CALCULER DES POLYNOMES

UN ANALOGUE ALGEBRIQUE DE LA FABULEUSE QUESTION P = ? NP

Plan du cours

1. Polynômes et monômes

Un polynôme est une somme de monômes.

Un monôme est un produit de la forme $c.x_1^{d_1}. \ldots . x_n^{d_n}$, où c est une constante non nulle, le coefficient du monôme, et les x_i des inconnues ; l'entier positif ou nul d_i est le degré du monôme en la variable x_i, la somme $d_1 + \ldots + d_n$ étant son degré total ; le uple d'entiers $(d_1, d_2, \ldots d_n)$ est son multidegré.

Les coefficients des polynômes se promènent dans un anneau commutatif K, qui pour nous sera le plus souvent un corps, ou bien l'anneau des entiers ; $K[x_1, \ldots$

$x_n]$, ou bien $K[\underline{x}]$, désigne l'anneau des polynômes en n variables $x_1, \ldots x_n$ et à coefficients dans K .

Dans l'écriture canonique d'un polynôme, on regroupe les monômes de même multidegré. Le degré total ou partiel d'un polynôme est le maximum des degrés de ses monômes.

Il convient de distinguer le polynôme nul, qui est sans monômes (une sommation indexée sur l'ensemble vide est nulle par convention) ; le polynôme nul n'a pas de degré (ou bien on convient de lui attribuer le degré -1). Un polynôme de degré nul, c'est donc une constante non nulle.

On peut additionner et multiplier des polynômes à coefficients dans K . C'est pour cela qu'ils forment un anneau ; on épargne ici au lecteur les pénibles vérifications de l'associativité du produit des polynômes, et de la distributivité de leur produit sur leur somme.

Lemme 1.1. *Si* K *est un corps (ou bien un anneau intègre), un polynôme de degré* d *en une variable, à coefficients dans* K *, n'y a pas plus de* d *zéros.*

Démonstration. Grâce au changement de variable $y = x - a$, on voit que tout polynôme $P(x)$ de degré $d > 0$ s'écrit $P(x) = Q(x).(x-a) + P(a)$, où $Q(x)$ est de degré $d-1$. Si a est un zéro de P , c'est que $x-a$ divise Q ; si b est un autre zéro de P , c'est un zéro de Q . La démonstration en suit par récurrence sur d . **Fin**

Corollaire 1.2. *Si* $A_1 , \ldots A_n$ *sont des parties infinies du corps* K *, le seul polynôme de* $K[\underline{x}]$ *nul sur* $A_1 \times \ldots \times A_n$ *est le polynôme nul.*

Démonstration. Par induction sur le nombre n de variables. D'après le Lemme 1.1, c'est vrai pour $n = 1$. Pour le passage de n à $n+1$, on considère un polynôme non nul $P(\underline{x},y)$ en $n+1$ variables, et on le développe par rapport à la variable y , le considérant comme polynôme à coefficients dans l'anneau $K[\underline{x}]$. Il s'écrit $P(\underline{x},y) = c_d(\underline{x}).y + \ldots + c_1(\underline{x}).y + c_0(\underline{x})$, avec un coefficient de tête $c_0(\underline{x})$ non identiquement nul. Par hypothèse de récurrence, on peut trouver a_1 dans A_1 , ... a_n dans A_n tels que $c(\underline{a}) \neq 0$, puis b dans A_{n+1} tel que $P(\underline{a},b) \neq 0$. **Fin**

Un polynôme $f(\underline{x})$ de $K[x_1, \ldots x_n]$ définit une application de $K^n = K \times \ldots \times K$ dans K ; une conséquence du Corollaire 1.2 est que, si K est un corps infini, on peut identifier un polynôme avec l'application qu'il définit. Ce n'est plus possible dans le cas d'un corps fini ; par exemple, les polynômes distincts x et x^2 définissent la même fonction sur le corps à deux éléments $F_2 = Z/2Z = \{0, 1\}$, leur différence $x^2 - x$ s'annulant en chaque point de ce corps.

L'addition et la multiplication des polynômes correspondent à celles des fonctions associées. Les applications-polynômes sont précisément les fonctions engendrées

par compositions successives à partir de l'addition, de la multiplication et des constantes.

Un polynôme à coefficients entiers définit une fonction de $K \times \ldots \times K$ dans K pour n'importe quel anneau K. Dit en termes pédants, cela vient de ce que le sous-anneau de K engendré par son unité est image homomorphe de l'anneau des entiers ; cet homomorphisme n'est pas nécessairement injectif, si bien qu'il y a quelque danger à identifier le 1 de l'anneau avec le 1 entier. Par exemple, si K est un corps, il peut contenir une copie de l'anneau Z des entiers, et par conséquent du corps Q des rationnels : on dit alors qu'il est de caractéristique nulle ; sinon, l'image homomorphe des entiers est de la forme $F_p = Z/pZ$ où p est un nombre premier, appelé caractéristique du corps. Nous considérerons souvent les polynômes à coefficients entiers modulo p, c'est-à-dire les polynômes à coefficients dans le corps Z/pZ.

Nous appellerons complexité du polynôme $f(\underline{x})$ le nombre d'additions (de deux objets) et de multiplications (de deux objets) qu'il faut pour le calculer (l'engendrer) à partir des constantes et des variables. Dans la plupart des cas, l'expression canonique du polynôme comme somme de monômes donne une méthode de calcul particulièrement maladroite (et dans tous les cas peu inventive !) ; par exemple $x^2 + 2.xy + y^2 = x.x + 2.x.y + y.y = (x+y)^2$ se calcule en deux opérations (additionner x et y, puis multiplier le résultat obtenu par lui-même), alors que son expression comme somme de monômes demande six opérations (quatre $.$ et deux $+$).

2. Déterminants et permanents

Nous considérons n^2 inconnues $x_{i,j}$, $1 \leq i \leq n$, $1 \leq j \leq n$, et les deux polynômes suivants :

$$Det_n(\underline{x}) = \Sigma_{\sigma \in Sn}\ \varepsilon(\sigma) . x_{1,\sigma 1} . \ldots . x_{n,\sigma n} = \Sigma_{\sigma \in Sn}\ \varepsilon(\sigma) . x_{\sigma 1,1} . \ldots . x_{\sigma n,n}$$

$$Per_n(\underline{x}) = \Sigma_{\sigma \in Sn}\ x_{1,\sigma 1} . \ldots . x_{n,\sigma n} = \Sigma_{\sigma \in Sn}\ x_{\sigma 1,1} . \ldots . x_{\sigma n,n} .$$

Formellement, ces deux polynômes se ressemblent beaucoup, puisque l'expression du permanent s'obtient en oubliant dans celle du déterminant la signature $\varepsilon(\sigma) = (+/-).1$ des permutations. Pourtant, alors que le déterminant se calcule facilement (nous allons voir comment), personne ne sait calculer le permanent, ni évaluer précisément sa complexité.

Il n'existe même aucune méthode connue de calcul efficace pour le problème numérique suivant, qui est associé au permanent. Une conjecture majeure de la Théorie de la complexité, c'est qu'il n'existe pas de méthode de calcul efficace pour ce problème, ce que personne n'a réussi à démontrer jusqu'à présent.

Ce problème est le problème des mariages. On considère des femmes, qui aiment des hommes ; plus précisément, on considère n femmes, qu'on dispose en colonnes, et n hommes qu'on met en lignes ; si la $i°$ femme aime le $j°$ homme, on met un 1 à l'intersection de la $i°$ ligne et de la $j°$ colonne, et sinon un 0. Le nombre total de façons de marier monogamiquement tous ces femmes, chacune à un homme qu'elle aime,et toutes ces femmes, est justement le permanent de cette matrice. Aucun dispositif connu ne permet de calculer ce nombre de mariages, disons pour $n = 10.000$; par contre, il est facile de voir si ce nombre est pair ou impair, puisque, modulo deux, $1 = -1$, et le permanent est égal au déterminant. Il est aussi facile de déterminer si le mariage global est possible, c'est-à-dire si ce permanent n'est pas nul. En effet :

FAUX **LEMME 2.01.** *Pour que le mariage soit possible, il faut et il suffit que, pour tout* $m \leq n$ *, il existe au moins* m *hommes qui soient aimés d'une femme parmi les* m *premières.*

Démonstration. La chose est bien sûr nécessaire. Nous montrons la réciproque par induction sur n ; c'est évident si $n = 1$. Pour $n > 1$, nous notons H_m l'ensemble des hommes qui sont aimés d'au moins une femme parmi les m premières f_1 , ... f_m , et nous distinguons deux cas. Dans le premier, pour tout $m < n$, $card(H_m) > m$; dans ce cas nous pouvons marier la première femme à un homme qu'elle aime, puisqu'elle en aime au moins deux, et si nous enlevons ce couple les $n-1$ femmes et les $n-1$ hommes qui restent satisfont au critère, et sont mariables par hypothèse d'induction. Dans le cas contraire, il existe $m < n$ tel que $card(H_m) = m$; on peut donc marier les m premières femmes avec les hommes de H_m (on n'a d'ailleurs pas le choix) ; les $m+k$ premières femmes aiment au moins $m+k$ hommes, et si on en enlève les m hommes de H_m qui sont aimés par certaines des m premières femmes, il reste au moins k hommes qui sont aimés par au moins une femme dans $\{f_{m+1}$, ... $f_{m+k}\}$; autrement dit les femmes et les hommes qui restent satisfont au critère, et sont mariables par hypothèse d'induction. **Fin**

Pour déterminer si le mariage est possible, il suffit donc de compter pour chaque m le nombre de 1 contenus dans les m premières lignes. On remarque que, quand c'est possible, il est facile de choisir en n étapes une façon de marier ces femmes et ces hommes, en suivant le pas de récurrence. On ne confondra pas un algorithme qui, à chaque pas, prend une décision adaptée à ce qu'il constate, avec un pseudo-algorithme qui à chaque pas doit explorer deux branches de possibilités ; notre algorithme est effectif, et n'a rien à voir avec la méthode totalement impraticable consistant à énumérer systématiquement les $n!$ associations de lignes et de colonnes jusqu'à ce qu'on en trouve une qui soit compatible avec la matrice. **Faux**

Evaluons le nombre d'opérations qu'il faut pour calculer le déterminant à partir de son expression comme somme de monômes (cette méthode était appelée "Règle de Sarrus" dans mon enfance mathématique) ; il y a $n!$ monômes à

ajouter, ce qui demande $n! - 1$ additions ; la moitié des monômes demandent un changement de signe, ce qui fait $n!/2$ multiplications par la constante -1 ; enfin, le calcul de chaque monôme demande $n-1$ multiplications d'inconnues. En tout, cela fait $n! - 1 + n!/2 + (n-1).n! = (n + 1/2).n! - 1$ opérations, ce qui est un nombre énorme, rendant le calcul totalement impraticable.

Pour le permanent, on est dispensé des changements de signes, si bien qu'il reste $n.n! - 1$ opérations, représentant un calcul tout aussi irréaliste.

Essayons l'itération du développement par rapport à une ligne ou une colonne, qui ramène le calcul d'un déterminant d'ordre n à celui d'une combinaison linéaire de n déterminants d'ordre $n - 1$; notons d_n le nombre d'opérations qu'il faut pour calculer un déterminant d'ordre n par cette méthode : $d_1 = 0$, $d_{n+1} = (n+1).d_n + n + n + 1 + [(n+1)/2]$. On pose $d_n = c_n.n!$, si bien que $c_{n+1} = c_n + (n + n + 1 + [(n+1)/2]) / (n+1)$, $c_{n+1} - c_n < 5/2n!$, et finalement $d_n < (5e/2).n!$, ce qui ne représente qu'une amélioration insignifiante du calcul précédent (d_n est de l'ordre de $n!$, puisque la série de terme général c_n converge). Le permanent, qui est lui aussi linéaire par rapport à ses lignes et à ses colonnes, se calcule de même ; comme il n'y a pas de signes à changer, le nombre d'opérations est majoré par $2e.n!$ (il est équivalent à $c.n!$ ou c est une constante positive inférieure à $2e$).

La méthode efficace de calcul du déterminant que nous allons examiner, c'est celle du pivot de Gauss, qui repose sur l'antisymétrie du déterminant (par rapport à ses lignes ou ses colonnes) : s'il y a deux lignes égales le déterminant est nul, ce qui n'est pas vrai pour le permanent.

3. Calcul du déterminant, avec des divisions.

Comme on calcule le déterminant en tant que polynôme, on n'a pas à se préoccuper de la nullité éventuelle du coefficient dans le coin pour le choix de la ligne pivot : on prend toujours la première ligne disponible.

Première étape : on annule tous les coefficients de la première colonne qui sont en dessous de $x_{1,1}$, en combinant chaque ligne avec la première, sans toucher cette dernière. Autrement dit, on remplace la $i°$ ligne L_i , $i > 1$, par $x_{1,1}.L_i - x_{1,i}.L_1$; cette manipulation multiplie le déterminant de la matrice par $(x_{1,1})^{n-1}$; pour chacune des $n-1$ lignes L_i à traiter, il faut un changement de signe, $2(n-1)$ multiplications et $n-1$ additions, puisqu'il est inutile de calculer le premier coefficient, qui est nul ; il faut en tout $(n-1).(3(n-1) + 1)$ opérations (on pourrait économiser des changements de signe en prenant les lignes opposées, dont le calcul ne demande que le changement de signe de $x_{1,1}$).

Ensuite on répète. Notons D_m le déterminant de la matrice obtenue après la (m-1)° étape, et d_m son coefficient (m,m), qui sera utilisé comme coefficient pivot à l'étape suivante. C'est ainsi que $D_1 = D$ est le déterminant à calculer, et $d_1 = x_{1,1}$. La m° étape de calcul demande $3.(n-m)^2 + (n-m)$, et multiplie le déterminant par $(d_m)^{n-m}$: $D_m = (d_m)^{n-m}.D_{m-1}$.

A la fin de l'itération, on aura effectué $\Sigma_1^{n-1} 3.x^2 + x = n^3 - n^2$ opérations. On obtient une matrice triangulaire, dont le déterminant D_n s'obtient en multipliant les n coefficients diagonaux d_i ; comme le déterminant a été multiplié par le produit des $(d_i)^{n-i}$, $i < n$, le déterminant D à calculer vaut le quotient de d_n par le produit des $(d_i)^{n-i-1}$, $i < n$; la formation de ce produit ne demande certainement pas plus de $\Sigma_1^{n-1} x = n(n-1)/2$ multiplications supplémentaires, et en fait beaucoup moins, car élever une quantité à la puissance m ne demande pas plus de log(m) +1 multiplications, ou log désigne le logarithme en base deux. En conclusion, nous calculons le déterminant avec moins de n^3 additions et multiplications, et une division.

A première vue, cette méthode paraît satisfaisante car elle n'exige pas un nombre exponentiel d'opérations. Cependant, on voit son défaut si on suit l'évolution des degrés des polynômes d_m, qui doublent à chaque fois, si bien qu'on finit par obtenir un polynôme d_n de degré exponentiel, égal à 2^{n-1}. Autrement dit, on exprime le déterminant, qui est un polynôme de degré n, comme une fraction rationnelle quotient d'un énorme numérateur par un énorme dénominateur, qui ont, bien sûr, beaucoup de facteurs communs.

Ce défaut devient très apparent si on tente une transposition numérique de la méthode. Soit à calculer un déterminant d'ordre n dont les coefficients sont des entiers qui, une fois écrits en base deux, n'ont pas plus de n chiffres. Nous vérifions tout d'abord que la question est sensée ; en effet, chaque coefficient a son module majoré par 2^n, si bien que celui du déterminant l'est par $(n!).(2^n)^n$, dont le logarithme est majoré par $n^2 + n.\log(n)$ puisque $n! \leq n^n$. En conséquence l'écriture de la valeur de notre déterminant ne demande qu'un nombre polynomial de chiffres, de l'ordre de n^2. Le problème, c'est que si on essaye de suivre numériquement la méthode ci-dessus, le nombre de chiffres des entiers manipulés double à chaque fois, et devient exponentiel avant la division finale : le calcul est totalement impraticable ! Il y a aussi un autre obstacle théorique qui, lui, est sans conséquences pratiques : quand on remplace les inconnues par des nombres dans un calcul de fractions rationelles, on peut tomber sur une division de 0 par 0, mais ce risque-là est tout-à-fait négligeable.

Pour éviter cette explosion des degrés, ou des tailles des nombres manipulés, nous améliorons la méthode de Gauss en introduisant des divisions à chaque étape, au lieu de les retarder en une seule division finale. On procède ainsi : effectuons les

deux premières étapes de la méthode de Gauss ; la matrice d'ordre trois qui est située dans le coin supérieur gauche est triangulaire, ayant d_1 , d_2 et d_3 sur sa diagonale ; le déterminant de cette matrice, qui vaut $d_1.d_2.d_3$, est égal au déterminant d'ordre trois placé dans le coin supérieur gauche de la matrice originelle multiplié par $d_1^2.d_2$, si bien que le quotient d_3/d_1 est égal à ce déterminant originel, et le polynôme d_3 est divisible par le polynôme d_1 ! Le même raisonnement vaut pour tous les déterminants d'ordre trois étendant le déterminant d'ordre deux situé dans le coin supérieur gauche, si bien que tous les coefficients des lignes à partir de la troisième sont divisibles par d_1 ; nous effectuons ces $(n-2)^2$ divisions. A l'étape suivante, nous gaussifions avec la troisième ligne, puis nous divisons tous les nouveaux coefficients par d_2 , ce qui fait $(n-3)^3$ divisions.

Si nous notons Δ_m la matrice obtenue après $m-1$ pas de la nouvelle méthode, et δ_m son coefficient (m,m) , le pas suivant consistera à pivoter sur la $m°$ ligne, puis à diviser tous les coefficients en-dessous par δ_{m-1} . Avant la division, on obtient un déterminant qui vaut $\delta_m^{n-m}.\Delta_m$, et après division $\Delta_{m+1} = \delta_m^{n-m}.\Delta_m / \delta_{m-1}^{n-m}$, si bien que $\Delta_{m+1} = \delta_1.\delta_2. \ldots \delta_{m-1}.\delta_m^{n-m}.D$. On voit donc que $\Delta_n = \delta_1.\delta_2. \ldots \delta_{n-1}.D$, si bien que $\delta_n = D$ est le déterminant à calculer ; en fait, δ_m est toujours égal au déterminant d'ordre m situé dans le coin supérieur gauche de la matrice originelle. Par cette méthode, le déterminant se calcule en moins de n^3 additions et multiplications et en $\Sigma_1^{n-2} x^2 < n^3/3$ divisions.

Son avantage, c'est que le degré des polynômes intermédiaires n'explose pas (ils sont tous de degré inférieur à $2n$), ce qui rend praticable, au moins en théorie, sa version numérique : les calculs intermédiaires ne font intervenir que des entiers s'écrivant avec un nombre polynomial de chiffres. Naturellement, la bonté de cet algorithme dépendra de la façon dont on effectue les multiplications et les divisions de nombres écrits en chiffres (il en existe de meilleures que celles de l'école primaire !).

Pour bien comprendre ce phénomène, on fera l'exercice suivant : on calcule avec une calculette un déterminant d'ordre quatre rempli de nombres de deux chiffres (en base dix !) pris au hasard ; si on ne fait que la division finale, après la troisième étape la calculette affiche le nombre d_4 en notation exponentielle, c'est-à-dire approximative, ayant dépassé ses possibilités d'affichage ; mais comme elle calcule avec plus de chiffres qu'elles n'en affiche, elle donne tout de même le résultat exact après la division de d_4 par $(d_1)^2.d_2$. Par contre, si on fait des divisions dès la deuxième étape, la calculette affiche toujours les nombres exacts.

4. Elimination des divisions

Comme il est peu satisfaisant pour l'esprit de devoir utiliser des divisions pour calculer un polynôme, nous allons nous en passer, au prix d'une légère augmentation du nombres d'additions et de multiplications à effectuer. La méthode, due à [STRASSEN 1973], repose sur le calcul des parties homogènes d'un polynôme, et sur la formule $1 + x + \ldots + x^d = 1 - x^{d+1} / 1 - x$.

La partie homogène de degré d d'un polynôme est par définition la somme de ses monômes de degré total d , et sa troncature au degré d est la somme de ses monômes de degré total inférieur ou égal à d . Nous convenons que la partie homogène de degré nul d'un polynôme f est son terme constant, même s'il est nul.

Lemme 4.1 (HOMOGENEISATION). *Si le polynôme* f *est de complexité* c *, sa partie homogène de degré* d *est de complexité au plus* $c.(d+1)^2$ *, et il en est de même de sa troncature au degré* d .

Démonstration. Dans un calcul du polynôme f en c étapes, nous considérons le polynôme f_i calculé à la $i°$ étape, et nous notons $f_{i,0}$, … $f_{i,d}$ ses parties homogènes de degré au plus d . Evaluons le nombre d'opérations qu'il faut pour calculer simultanément toutes ces parties homogènes.

Les parties homogènes des variables et des constantes sont immédiatement calculées.

Si f_i est obtenu comme somme $f_i = f_i' + f_i''$ de deux polynômes, éventuellement confondus, antérieurement calculés, $f_{i,j}$ est la somme de f_i',j et de f_i'',j . Autrement dit, une addition dans le calcul de f se transforme en $d+1$ additions dans le calcul de ses parties homogènes.

Si f_i est obtenu comme produit $f_i = f_i'.f_i''$ de deux polynômes, antérieurement calculés, $f_{i,j} = \Sigma_0^j \; f_i',k.f_i'',j-k$, ce qui demande j additions et $j+1$ multiplications. Autrement dit, une multiplication dans le calcul de f se transforme en $\Sigma_0^d \; 2j+1 = (d+1)^2$ opérations dans le calcul de ses parties homogènes.

En conséquence, le calcul des parties homogènes multiplie la complexité du calcul par au plus $(d+1)^2$. Si le calcul de f ne comprend que des multiplications, ce dernier est un monôme ; sa troncature, étant nulle ou égale à f , ne demande pas plus de c opérations. S'il faut au moins une addition, le calcul des parties homogènes de f ne demande pas plus de $(c-1).(d+1)^2 + d + 1$ opérations ; pour obtenir la troncature, il faut ensuite les sommer, ce qui demande d opérations de plus ; d'où le résultat, puisque $2d + 1 < (d+1)^2$. **Fin**

Lemme 4.2 (INVERSION). *Si* f *est un polynôme de degré au plus* d *, qui est divisible par le polynôme* $1 - g$ *, où* g *est sans terme constant, alors le polynôme* $f / 1-g$ *est égal à la troncature à l'ordre* d *du polynôme* $f.(1 + g + \ldots + g^d)$.

Démonstration. En multipliant l'équation $(1 - g).(1 + g + \ldots + g^d) = 1 - g^{d+1}$ par $f / 1-g$, on obtient l'identité polynomiale $f / 1 - g = f.(1 + g + \ldots + g^d) + g^{d+1}.(f / 1 - g)$; par ailleurs, le polynôme $f / 1 - g$ est de degré au plus d , tandis que g^{d+1} a tous ses monômes de degré strictement supérieur à d . **Fin**

On perfectionne d'une autre manière la méthode de Gauss (à division finale). Il nous faut tout d'abord déshomogénéiser le déterminant : on introduit de nouvelles inconnues $y_{i,j}$, et on calcule le déterminant de coefficients $1 + y_{i,i}$ sur la diagonale, et $y_{i,j}$, $i \neq j$, ailleurs. Pour obtenir ce qu'on cherche, il suffira à la fin de remplacer $y_{i,i}$ par $x_{i,i} - 1$ et $y_{i,j}$ par $x_{i,j}$.

Tous les d_m ont 1 comme coefficient constant, puisque si on annule toutes les inconnues on calcule le déterminant de la matrice identité, et que la méthode du pivot ne modifie en rien cette matrice (chaque étape consiste à multiplier des lignes par 1 et à leur ajouter la ligne nulle !). On pose $\Pi_{1 \leq i < n-1} (d_i)^{n-i-1} = 1 - g$, et au lieu de diviser d_n par $1-g$, on le muliplie par $1 + g + \ldots + g^n$. Le nombre d'additions et de multiplications nécessaires reste inférieur à n^3 . Il faut ensuite tronquer le calcul au degré n , ce qui multiplie la complexité par $(n+1)^2$.

En conclusion, on peut calculer un déterminant d'ordre n comme polynôme, c'est-à-dire en n'utilisant que des additions et des multiplications, le nombre total de ces opérations étant de l'ordre de n^5 , et même majoré par $n^3.(n+1)^2$. Dans ce calcul, tous les polynômes intermédiaires sont de degré total inférieur ou égal à n .

Pour les applications numériques, cette méthode est moins bonne que la précédente, puisque multiplier ou diviser deux nombres écrits en binaire sont des opérations de coûts équivalents. On voit que, face à un problème numérique, l'algorithme le plus astucieux n'est pas toujours de résoudre le problème "en lettres", puis de remplacer les inconnues par leur valeur.

5. Dériver un polynôme

Si f est un polynôme de complexité c , que peut-on dire de la complexité de sa dérivée f' par rapport à l'une de ses variables ? Eh bien, elle est inférieure à $4.c$. En effet, à partir d'un calcul de f en c opérations, nous essayons de conduire simultanément celui des dérivées des polynômes intermédiaires obtenus, et de

compter le nombre total d'opérations nécessaires ; dériver une constante ou une variable ne coûte rien ; si le polynôme f_i est une somme $f_j + f_k$, alors $f_i' = f_j' + f_k'$, et il faut une addition de plus ; si le polynôme f_i est un produit $f_j.f_k$, alors $f_i' = f_j.f_k' + f_k.f_j'$, et il faut une addition et deux produits de plus ; dans le pire des cas, le nombre d'opérations est multiplié par quatre.

Cette simple remarque illustre ce que doit savoir d'instinct tout étudiant de première année de nos universités : quand on demande de dériver une expression où apparaissent des produits, il ne faut surtout pas la développer !

On peut voir que si on dérive un polynôme n fois par rapport à la même variable, la complexité n'explose pas (voir [KS 1991]). Le bon algorithme de dérivation n fois ne consiste pas, bien sûr, à répéter n fois l'algorithme de dérivation une fois !

Par contre, il est probable que des dérivations itérées par rapport à des inconnues distinctes fassent exploser les calculs. Considérons en effet n^2 inconnues $x_{i,j}$, n inconnues y_j , et le polynôme $f(\underline{x},\underline{y})$ produit des $x_{i,1}.y_1 + x_{i,2}.y_2 + \ldots + x_{i,n}.y_n$, pour $1 \leq i \leq n$; son calcul demande $n(2n-1) + n-1 = 2n^2 - 1$ opérations ; quand on le développe comme polynôme en $\underline{y}$, le coefficient du monôme $y_1.y_2. \ldots y_n$ est le permanent de $\underline{x}$, et dans tous les autres monômes il manquera un y_i (et un autre y_j figurera à la puissance au moins deux). Tout ceci fait que $\partial^n f / \partial y_1.\partial y_2. \ldots .\partial y_n = Per_n(\underline{x})$, un polynôme présumé difficile à calculer.

On remarque en passant qu'un polynôme simple peut avoir des coefficients compliqués.

On peut aussi essayer d'intégrer des polynômes. En dépit de l'intégration par parties, il n'y a pas de formule qui donne la primitive de $f.g$ en fonction de f , de g , et de leurs primitives, si bien que le fait que la primitive d'un polynôme soit elle aussi un polynôme est une espèce de miracle dont on ne se rend compte qu'en le développant en monômes : c'est pour cela qu'il est bien plus pénible d'intégrer que de dériver. Suivant ce principe, une intégrale simple devrait à elle seule faire exploser la complexité des calculs (elle pose un problème de coefficients, car la primitive d'un polynôme à coefficients entiers n'est pas un polynôme à coefficients entiers), mais il n'y a pas d'exemples probants connus.

Cet exemple devrait être de grand degré par rapport à sa complexité ; en effet, si $p(\underline{x},y)$ est un polynôme de complexité c et de degré d , par homogénéisation partielle, par rapport à la variable y , on peut calculer en moins de $c.(d+1)^2$ opérations tous les monômes $a_0(\underline{x})$, $a_1(\underline{x}).y$, … $a_d(\underline{x}).y^d$ de $p(\underline{x},y) = a_0(\underline{x}) + a_1(\underline{x}).y + \ldots + a_d(\underline{x}).y^d$ qui apparaissent dans son développement comme polynôme en y ; en multipliant le $i°$ monôme par la constante $1/i+1$ (pour ce type de calcul, on doit se donner gratuitement les constantes rationnelles), en

sommant tous ces monômes et en multipliant le résultat par y on obtient le polynôme $\int_0^y p(\underline{x},u).du$, dont la complexité est en conséquence majorée par $c.(d+1)^2 + 2.d + 1$. En fait, en traitant à part le cas où p est un monôme, on peut majorer cette complexité par $c.(d+1)^2$, puisque $3.d + 1 \leq (d+1)^2$.

Voici par contre un exemple d'intégrale multiple ; on pose $g(\underline{x},\underline{y}) = y_1.y_2. \ldots .y_n.f(\underline{x},\underline{y})$; tous les monômes de g sauf $(y_1.y_2. \ldots .y_n)^2$ ont une inconnue qui intervient au premier degré, si bien que $\int_{-1}^{1}\int_{-1}^{1} \ldots \int_{-1}^{1} g(\underline{x},\underline{y}).dy_1.dy_2. \ldots .dy_n = (2/3)^n.Per_n(\underline{x})$.

Les exemples de cette section sont extraits de [BÜRGISSER 2000].

6. Sleuppes et circuits

Le moment est venu d'introduire des objets, de nature combinatoire, permettant de définir de façon plus rigoureuse, ou plus imagée, la complexité d'un polynôme.

Un sleuppe (SLP = straight line program en anglo-américain), c'est une suite finie de polynômes $f_1 , \ldots f_n$, telle que pour tout i , f_i soit une variable, ou bien une constante, ou bien la somme de deux polynômes antérieurs ($f_i = f_j + f_k$, $j < i$, $k < i$; il est possible que $j = k$), ou bien le produit de deux polynômes antérieurs. Le dernier polynôme f_n est le polynôme calculé par le sleuppe ; la complexité du sleuppe, c'est le nombre de f_i qui correspondent à une opération, c'est-à-dire qui ne sont ni une variable, ni une constante. On définit alors la complexité d'un polynôme comme la complexité minimale d'un sleuppe le calculant, ce qui correspond bien à la notion informelle que nous avons employée jusqu'à présent.

Un multicircuit arithmétique est un dispositif plus élaboré qu'un sleuppe, bien qu'il définisse la même notion de complexité. Il s'agit d'un graphe fini orienté, sans cycles orientés, dont les arêtes sont appelées flêches, et dont les sommets, appelées portes, sont indexés par certains symboles, satisfaisant aux conditions suivantes :

- les portes qui ne reçoivent aucune flêche, qui sont appelées entrées, sont étiquetées par une variable, ou bien par une constante ;
- les autres portes reçoivent exactement deux flêches, de la même porte ou bien de deux portes distinctes ; elles sont étiquetées soit par le symbole d'addition, soit par le symbole de multiplication.

Nous appellerons circuit un multicircuit qui n'a qu'une seule sortie, c'est-à-dire qu'une seule porte n'émettant pas de flêches.

La taille d'un multicircuit, c'est son nombre total de portes, et sa complexité est son nombre de portes qui ne sont pas des entrées.

On appelle chemin orienté allant de la porte p à la porte q une suite finie de flêches $(f_1, f_2, \ldots f_n)$ telle que p soit la source de f_1, q soit le but de f_n, et pour chaque i, $1 \leq i < n$, le but de f_i soit la source de f_{i+1} ; on convient que chaque porte est reliée à elle-même par le chemin vide.

Si C est un multicircuit et p une de ses portes, le sous-circuit $C(p)$ associé à cette porte est formé de toutes les portes q pour lesquelles il existe dans C un chemin orienté allant de q à p.

On définit ainsi, par induction sur la taille, le polynôme calculé à une porte p du multicircuit C, soit encore le polynôme calculé par le sous-circuit $C(p)$: pour une entrée, c'est la constante ou la variable qui l'étiquette ; pour une porte-addition, c'est la somme des polynômes calculés aux deux portes (éventuellement confondues) dont elle reçoit ses flêches ; pour une porte-multiplication, c'est le produit des polynômes calculés à ces deux portes. Un circuit calcule un polynôme, celui qu'on obtient à sa sortie, tandis qu'un multicircuit à m sorties calcule un m-uple de polynômes.

Il est immédiat de transformer un sleuppe en un circuit calculant le même polynôme : les portes du circuit sont les f_i ; si le polynôme f_i est variable ou constante, on en fait une entrée d'étiquette correspondante ; si $f_i = f_j + f_k$, on étiquette la $i°$ porte par $+$ et on met une flêche de j vers i et une flêche de k vers i ; on fait de même pour un produit ; comme $j < i$ et $k < i$, il n'y a pas de cycles. Ensuite, on jette toutes les portes qui n'interviennent pas dans le calcul de f_n, de sorte qu'il n'y ait qu'une seule sortie (si on ne fait pas ça, on obtient un multicircuit).

Pour transformer un circuit en sleuppe, il suffit d'ordonner totalement ses portes de manière que chacune ne puisse recevoir de flêches que de portes antérieures ; c'est possible précisément parce que le graphe est sans cycles (toute relation binaire sans cycles se renforce en un ordre total).

Circuits et sleuppes définissent donc la même complexité pour les polynômes. L'avantage des circuits, c'est qu'ils permettent de lire naturellement certains paramètres significatifs associés à un calcul, si bien que nos démonstrations seront le plus souvent des manipulations de circuits.

Une propriété essentielle des circuits, c'est qu'une porte peut émettre autant de flêches qu'elle veut (pourvu, bien sûr, qu'elle trouve suffisamment d'autres portes pour les recevoir !) ; cela signifie qu'un calcul, une fois fait, peut être utilisé autant de fois qu'on veut. Parmi les circuits, nous distingons les termes, qui sont ceux dans lesquels toutes les portes, sauf la sortie, émettent une flêche et une seule : ils modélisent les situations où, si on a besoin d'utiliser un calcul une deuxième fois, il faut le repéter ; cela semble a priori une contrainte algorithmique bizarre et peu

naturelle, mais les termes sont importants car ils interviennent dans la représentation des calculs parallélisés, comme nous allons le voir.

Nous appellerons expression arithmétique un terme dont les portes sont étiquetées seulement par des variables, qui en outre sont toutes distinctes ; un terme s'obtient à partir d'une expression de même complexité par substitution de variables ou de constantes aux variables de cette expression.

Nous avons donc plusieurs façons d'écrire un même polynôme :

- canoniquement, comme somme de monômes : x.x + y.y + 2.x.y + x.z + y.z , représentant un calcul de dix opérations (six . et quatre +)
- comme terme : (x + y).((x+y) + z) , quatre opérations
- comme sleuppe, aisément transformable en circuit : x , y , z , u = x+y , v = u + z , u.v , trois opérations.

On voit que la taille des termes, qui sont pourtant la façon habituelle d'écrire les formules en Logique, ne représente pas fidèlement la complexité des polynômes qu'ils calculent ; l'écriture canonique comme somme de monômes est bien sûr encore pire. La façon dont nous avons calculé un déterminant en à peu près n^5 opérations se représente par un circuit de même taille ; il s'agit bien d'un circuit, et non d'un terme, car les coefficients pivots d_m sont utilisés plusieurs fois en cours de calcul.

Par contre, la complexité exponentielle que nous avons trouvée pour le développement itéré par rapport à une ligne est une complexité termique. En fait, il n'est pas connu si un déterminant d'ordre n peut être calculé par un terme arithmétique de taille polynomiale en n : on pense plutôt que non, mais c'est encore un problème ouvert en théorie de la complexité.

Voici un lemme évident, mais néanmoins utile, caractérisant les termes :

Lemme 6.1 (CARACTERISATION DES TERMES). *Un circuit* C *est un terme si et seulement si chaque porte* q *de* C *est relié par un chemin et un seul à la sortie, si et seulement si pour chacune de ses portes d'opération* p *, recevant ses flêches de* p' *et de* p" *, les deux sous-circuits* C(p') *et* C(p") *sont disjoints.*

Démonstration. Si C est un terme, il est clair qu'il n'y qu'un seul chemin partant d'une porte q , et qui finit par aboutir à la sortie ; il est alors impossible que C(p') et C(p") aient une porte commune q , car on trouverait un chemin passant par p' , et un autre passant par p", partant de q et menant à la sortie.

Réciproquement, si C n'est pas un terme, il a une porte q qui émet au moins deux flêches, qui sont le point de départ de deux chemin distincts, qui doivent se rejoindre au plus tard à la sortie, en une porte p avec C(p') et C(p") non disjoints. **Fin**

Examinons maintenant le lien entre la complexité et le nombre d'entrées.

Lemme 6.2 (TAILLE ET COMPLEXITE). *Un circuit de complexité* c *a au plus* c+1 *entrées (et sa taille vaut donc au plus* 2.c + 1 *). Il a* c+1 *entrées si et seulement si c'est un terme.*

Démonstration. Comme chacune des c portes d'opération du circuit reçoit deux flêches, le circuit comporte en tout 2.c flêches ; s'il y a e entrées, chacune des e + c – 1 portes qui ne sont pas la sortie doit émettre au moins une flêche, si bien que $e + c - 1 \leq 2.c$, soit encore $e \leq c + 1$. Il y a égalité précisément quand chacune de ces portes n'a droit qu'à une flêche. **Fin**

A un circuit C , nous associons, outre sa taille et sa complexité, un certain nombre de paramètres numériques. Le premier est sa profondeur p(C) , qui est la longueur maximale d'un chemin orienté joignant une entrée à la sortie ; la profondeur est majorée par la complexité. Bien souvent, on cherche à équilibrer un calcul de manière à minimiser sa profondeur (la profondeur représente le "temps parallèle" nécessaire au calcul).

Voici un lemme trivial de remplacement d'un circuit par un terme, sans changement de profondeur, mais avec explosion exponentielle de complexité, si elle est proche de la profondeur.

Lemme 6.3 (PROFONDEUR). *Un circuit* C *de profondeur* p *peut être remplacé par un terme de même profondeur, de complexité au plus* 2^p - 1 *, qui calcule, avec les mêmes constantes, le même polynôme.*

Démonstration. Par induction sur p ; si p = 0 , le circuit est réduit à une entrée, qui est un terme de profondeur et de complexité nulles ; si $p = q + 1$, la sortie s du circuit reçoit ses deux flêches de portes s' et s" , éventuellement confondues, dont les sous-circuits associés sont de profondeur au plus q ; C(s') et C(s") s'exprimant comme des termes de complexité au plus $2^q - 1$ et de profondeur q , C s'exprime comme terme de profondeur $q + 1 = p$ et de complexité au plus $2^q - 1 + 2^q - 1 + 1 = 2^p - 1$. **Fin**

La même induction montre que la complexité de C est majorée par $2^p - 1$. Il est par ailleurs évident que la complexité majore la profondeur.

A un circuit C , nous associons également l'entier a(C) , que nous appelerons son ampleur, qui est calculé par le circuit C_a obtenu à partir de C en remplaçant toutes les étiquettes d'entrées par la constante 1 . Si le circuit n'utilise pas de constantes, ou bien seulement la constante 1 , son ampleur est tout simplement la valeur en (1, ... 1) du polynôme qu'il calcule. Ce nombre est associé aux monômes unitaires du circuit qui sont ainsi définis : un monôme unitaire est un monôme de coefficient 1 ; un polynôme calculé par un circuit arithmétique sans paramètres est somme de monômes unitaires, ses monômes véritables étant obtenus en regroupant les monômes unitaires de même multidegré.

Lemme 6.4 (MONOMES UNITAIRES). *Soit* f *le polynôme calculé par le circuit* C *. S'il n'a pas de constantes,* f *est somme de* a(C) *monômes unitaires. Si* -1 *est la seule constante utilisée,* a(C) *majore la somme des valeurs abolues des coefficients des monômes de* f .

Démonstration. Les nombres de monômes unitaires s'ajoutent quand on franchit une porte d'addition, et se multiplient quand on franchit une porte de multiplication. Si 1 est la seule constante utilisée, a(C) vaut donc la somme des coefficients des monômes, tandis que, si on utilise -1 , il y a des changements de signes. **Fin**

Nous introduisons aussi le sous-degré sd(C) d'un circuit, ou plutôt de ses portes, par induction sur la profondeur, de la manière suivante : le sous-degré d'une constante vaut 0 , et celui d'une variable vaut 1 ; le sous-degré d'une porte d'addition est le maximum des sous-degrés des portes dont elle reçoit ses flêches, et celui d'une porte de multiplication est leur somme ; le sous-degré du circuit est celui de sa sortie. Ce sous-degré se calcule rapidement par un circuit numérique, où les portes d'addition sont remplacées par des prises de maximum et les portes de multiplication par des additions.

Il est clair qu'un circuit de sous-degré d calcule un polynôme de degré au plus d ; si on utilise la constante -1 , il peut y avoir des simplifications de monômes, si bien que le degré réel du polynôme calculé peut être bien inférieur au sous-degré du circuit. D'un autre côté, si un circuit C de complexité c calcule un polynôme f qui se trouve être de degré d , f est égal à sa troncature au degré d , qui, nous l'avons vu, se calcule par un circuit C' de complexité inférieure à $c.(d+1)^2$ et de sous-degré d .

Cette notion de sous-degré a le défaut de traiter différemment les variables et les constantes, si bien que nous perdons sa trace quand nous nous permettons de remplacer des constantes par des variables. Pour cette raison, nous introduisons le degré d(C) d'un circuit défini de la même façon que son sous-degré, sauf que, cette fois, les constantes sont elles-aussi de degré un. Remarquons en passant que, si le circuit C n'utilise pas de constantes, d(C) est exactement le degré du polynôme calculé par C .

En présence de constantes, nous allons devoir affronter une difficulté : le Lemme de troncature parle du sous-degré, tandis que c'est le degré qui intervient dans les manipulations de circuits, le sous-degré étant un invariant peu satisfaisant.

Nous sommes maintenant armés pour démontrer des lemmes évaluant l'influence des paramètres des circuits sur les polynômes qu'ils calculent.

Lemme 6.5 (CONTRAINTES SUR UN POLYNOME CALCULE PAR UN TERME). *Un terme de complexité* c *calcule un polynôme en au plus* c+1 *variables, de degré total au plus* c+1 *; ce polynôme a moins de* 2^c *monômes ; si la seule constante utilisée dans le calcul est* -1 *, les coefficients de ces monômes sont des entiers de valeur absolue inférieure à* 2^c .

Démonstration. Nous avons vu que ce terme a $c+1$ entrées, si bien que le polynôme qu'il calcule ne dépend que d'au plus $c + 1$ variables. Comme le maximum de deux nombres est toujours inférieur à leur somme, le degré de T est inférieur à celui du terme T' obtenu à partir de T en remplaçant toutes les portes d'opérations par des multiplications. Comme T' calcule le produit de ses entrées, son degré vaut $c + 1$.

Pour majorer le nombre de monômes, ainsi que les valeurs absolues des coefficients, on montre, par induction sur c , que $a(T)$ est inférieur à 2^c . C'est clair si $c = 0$. Sinon le terme s'écrit $T(s').T(s")$, ou bien $T(s') + T(s")$, où s' et $s"$ sont les deux portes juste avant la sortie ; si on note c' et $c"$ les complexités de ces sous-termes disjoints, on voit que $c = c' + c" + 1$. Par hypothèse d'induction, leurs amplitudes respectives sont majorées par $2^{c'}$ et $2^{c"}$. Dans le premier cas, $a(T) \leq 2^{c'}.2^{c"} < 2^c$; dans le deuxième $a(T) \leq 2^{c'} + 2^{c"} \leq 2.2^{\mathrm{Max}(c', c")} \leq 2^c$. **Fin**

Lemme 6.6 (CONTRAINTES SUR UN POLYNOME CALCULE PAR UN CIRCUIT). *Un polynôme calculé par un circuit de taille* c *et de profondeur* p *calcule un polynôme en au plus* $c+1$ *variables, dont le degré total est majoré par* $2^p \leq 2^c$ *; il a moins de* $2^{p.c} \leq 2^{c\hat{}2}$ *monômes ; si la seule constante utilisée dans le calcul est* -1 *, les coefficients de ces monômes sont des entiers de valeur absolue inférieure à* $2\hat{}(2^p) \leq 2\hat{}(2^c)$.

Démonstration. Nous avons vu que le circuit n'a pas plus de $c+1$ entrées.

On majore le degré du circuit par induction sur sa profondeur. Si $p = 0$, $d(C) = 1$; sinon, les deux sous-circuits antérieurs à la sortie ont leur degré majoré par 2^{p-1} , qui reste un majorant du degré de la sortie si elle est additive ; si elle est multiplicative, son degré est au plus $2^{p-1} + 2^{p-1} = 2^p$. Il est par ailleurs clair que la complexité majore la profondeur.

On remarque que le degré 2^p n'est possible que si on ne fait que des multiplications, et qu'il n'y a alors qu'un seul monôme ; par ailleurs, s'il y a $c+1$ inconnues le circuit est un terme, dont nous avons déja majoré le nombre de monômes. Sinon, avec c variables on peut former $(2^p)^c = 2^{p.c} \leq 2^{c\hat{}2}$ monômes de degré au plus $2^p - 1$ par rapport à chacune des variables, ce qui est plus que le nombre de monômes de degré total strictement inférieur à 2^p .

Pour majorer les coefficients, on montre par induction sur p que $a(C)$ est (strictement) inférieur à $2\hat{}(2^p)$. **Fin**

Autrement dit un polynôme exprimable comme un terme de complexité c a un nombre de variable et un degré polynomiaux en c, et si le calcul n'utilise que -1 comme constante les coefficients des monômes sont des entiers qui ne demandent pour s'écrire qu'un nombre polynomial de chiffres. Pour un circuit de complexité c, le degré et le nombre de chiffres deviennent exponentiels, c'est-à-dire que les coefficients deviennent doublement exponentiels. Dans les deux cas, le nombre de monômes reste simplement exponentiel (i.e. exponentielle d'un polynôme en c), mais pour des raisons différentes : dans le cas des termes, on controle vraiment le nombre de monômes unitaires qui apparaissent en cours de calcul ; dans le cas des circuits les plus généraux, on constate seulement que les contraintes sur le nombre de variables et le degré n'autorisent qu'un nombre simplement exponentiel de multidegrés.

Il n'y a aucun espoir que ce type de contraintes puisse caractériser, même approximativement, les polynômes calculables par un terme ou un circuit de complexité c, car ils y a beaucoup plus de polynômes qui les respectent que de termes ou de circuits de cette complexité.

Pour nous résumer, nous avons montré que la complexité c, la profondeur p, l'ampleur a et le degré d d'un circuit satisfont aux conditions suivantes : $p \leq c$, $c \leq 2^p - 1$, $d \leq 2^p \leq 2^c$, $a < 2\hat{}(2^p) \leq 2\hat{}(2^c)$; dans le cas d'un terme, nous avons obtenu aussi $d \leq c+1$ et $a \leq 2^c$. A ces contraintes, nous en ajoutons une dernière :

Lemme 6.7 (AMPLEUR ET DEGRE). *Quel que soit le circuit* C*, de complexité* c*, de degré* d *et d'ampleur* a*,* $a \leq 2^{cd}$ *; si en outre aucune entrée de* C *n'envoie de flèche vers une porte de multiplication, et* $c > 0$*,* $2^d \leq a$.

Démonstration. Par induction sur c. Si $c = 0$, $a = 1$. Si $c > 0$, et la sortie est une addition, $a \leq 2^{(c-1)d} + 2^{(c-1)d} = 2^{cd-d+1} \leq 2^{cd}$ puisque $d \geq 1$. Si $c > 0$ et la sortie est une multiplication, $a \leq 2^{(c-1)d'}.2^{(c-1)d''} < 2^{c(d'+d'')} = 2^{cd}$.

On montre la minoration de a par induction sur la profondeur ; si la sortie est une addition qui reçoit ses flèches de deux entrées, $a = 2$ et $d = 1$; si la sortie est une addition dont au moins une antécédente n'est pas une entrée, $a > 2^d$; si la sortie est une multiplication, $a \geq 2^{d'}.2^{d''} = 2^{d'+d''} = 2^d$. **Fin**

Si on appelle biampleur b du circuit C le nombre qu'il calcule lorsqu'on remplace toutes ses entrées par 2, on peut montrer que $2^d \leq b \leq 2^{(c+1)d}$.

7. Parallélisation

Les contraintes pesant sur les polynômes calculés par les termes et les circuits font pressentir une corrélation logarithmique entre la taille des termes et la profondeur des circuits.

Comme nous l'avons vu, un circuit de profondeur p se remplace par un terme de même profondeur calculant le même polynôme (avec les mêmes constantes), obtenu en disjoignant à chaque niveau en commençant par sa sortie ; la complexité de ce terme est majorée par $2^p - 1$. Autrement dit, si ce circuit est de profondeur $A.\log(n)$, la complexité du terme sera majorée par n^A ; le lemme suivant affirme une sorte de réciproque, à savoir que si le circuit peut-être remplacé par un terme de complexité inférieure à n^A , il peut être ré-écrit en profondeur $3A.\log(n) + 1$.

Théorème 7.1 (PARALLELISATION DES TERMES ARITHMETIQUES, [MP 1976]). *Un terme de complexité* $c > 0$ *peut se remplacer par un terme de profondeur majorée par* $3.\log(c) + 1$ *, et donc de complexité* $2.c^3$ *à peine plus grande, qui calcule, avec les mêmes constantes, le même polynôme.*

Démonstration. Nous pouvons supposer que le terme est une expression (pas de constantes, les entrées sont indexées par des variables distinctes), puisque tout terme s'obtient à partir d'une expression par substitution de variables ou de constantes ; dans une expression, complexité et nombre de variables sont rigidement associés : si la complexité est c , le nombre de variables est $c + 1$.

On fait l'observation préliminaire suivante : si $f(\underline{x},y)$ est calculé par une expression, dans laquelle on met de côté la variable y , il s'écrit sous la forme $f(\underline{x},y) = A(\underline{x}').y + B(\underline{x}")$, où $A = 1$ ou bien est une expression, et où $B = 0$ ou bien est une expression. Cette écriture n'est pas une expression, car les uples de variables $\underline{x}'$ et $\underline{x}"$ ne sont pas nécessairement disjoints. Cela se montre sans difficulté par induction sur la complexité de l'expression.

L'induction s'amorce facilement pour $c = 1$. Pour les plus grandes valeurs de c , on choisit un sous-terme qui contient à peu près la moitié des variables ; plus précisément, on choisit dans notre expression une sous-expression minimale E qui soit de complexité strictement supérieure à $c/2$: $E = F*G$, où $*$ est une multiplication ou une addition, et où F et G sont de complexité inférieure ou égale à $c/2 < c$; par hypothèse d'induction, elle se remplacent par des termes (pas des expressions !) F' et G' de profondeur au plus $3.\log(c/2) + 1$: cela est vrai même si ce sont des variables, de complexité et de profondeur nulle, puisque $c \geq 2$. Considérons alors l'expression $H(\underline{x},y)$ obtenue en remplaçant dans l'expression originelle la sortie de E par une nouvelle variable y ; comme la complexité de E est strictement supérieure à $c/2$, celle de H lui est strictement inférieure ; H se met sous la forme $A(\underline{x}').y + B(\underline{x}")$, où A et B , si ce ne sont pas les constantes 1 ou 0 , sont des expressions qui ont une variable de moins, donc qui sont de complexité moindre, et par hypothèse d'induction se parallélisent en A' et B' de profondeur au plus $3.\log(c/2) + 1$. L'expression originelle

équivaut à $A'.(F'*G') + B'$, qui est de profondeur majorée par $3.\log(c/2) + 1 + 3 = 3.\log(c) - 3 + 4 = 3.\log(c) + 1$. **Fin**

Par contre, on ne sait pas si le calcul des polynômes respectant les contraintes décrites dans la section précédente peut être parallélisé sans explosion de complexité. Plus précisément, on ne sait pas s'il existe un polynôme $p(x,y,z)$ à coefficients réels, tel que tout polynôme f , de degré d , à coefficients entiers de modules majorés par 2^n , et calculé par un circuit de complexité c n'utilisant que la constante -1 , se calcule par un terme de complexité inférieure à $p(d,n,c)$; d'après le Lemme de parallélisation, une question équivalente est de demander un circuit de profondeur au plus $\log(q(d,n,c))$ pour un certain polynôme q . On pense que c'est faux - et il est même possible que ce soit faux pour les déterminants - mais on ne sait pas le démontrer.

Et, en effet, les deux théorèmes suivants ne sont que des semi-parallélisations, puisque les profondeurs obtenues sont polylogarithmiques.

Théorème 7.2 (SEMI-PARALLELISATION DES CIRCUITS, [VSBR 1983]). *Un circuit* C , *de complexité* $c > 0$ *et de degré* d *, se remplace par un circuit équivalent de profondeur* $\log(4d).\log(8c)$ *, de complexité* $6.d^{\log(3)}.c^3$ *, de degré* $3.d^{\log(3)}$ *et de même ampleur, qui calcule le même polynôme, et qui utilise la constante* 1 *en plus des constantes de* C .

Démonstration. On peut supposer que toutes les entrées de C sont étiquetées par des variables. Dans un premier temps, nous ne nous occupons que de borner la profondeur du calcul. Il alors suffit de montrer, par récurrence sur m , que si $2^{m-1} < d \leq 2^m$ ce circuit se parallélise en profondeur $(m+1).\log(8c)$, puisque m est majoré par $\log(d) + 1 = \log(2d)$. Si $m = 0$, $d = 0$, C ne contient que des additions, et se parallélise en profondeur $\log(c) + 1 = \log(2c)$.

Si $m > 0$, nous notons p_1 , ... p_k les portes minimales dans C de degré strictement supérieur à 2^{m-1} ; ce sont des portes de multiplication ; nous notons p_i' et p_i'' les portes qui envoient leurs flèches vers p_i .

Soit B le sous-multicircuit de C dont les sorties sont les p' et les p'' ; ces dernières portes ont une complexité inférieure à c , et un degré inférieur ou égal à 2^{m-1} , si bien que l'hypothèse d'induction s'applique ; on peut donc paralléliser B en profondeur $m.\log(8c)$.

Notons A le circuit obtenu en remplaçant p_1 , ... p_k par des entrées indexées par des variables y_1 , ... y_k , en supprimant pour chacune d'entre elles les deux flèches qu'elle reçoit, et en éliminant les portes qui ne sont plus sur un chemin arrivant à la sortie de C . La sortie de A calcule un polynôme au plus du premier degré en y_1 , ... y_k , de la forme $a_1(\underline{x}).y_1 + ... + a_k(\underline{x}).y_k + a_0(\underline{x})$. Les polynômes $a_1(\underline{x})$, ... $a_k(\underline{x})$ sont de degré au plus $2^{m-1} - 1$. Les monômes $a_i(\underline{x}).y_i$ sont calculés par des circuits A_1 , ... A_k de complexité inférieure à c ; en effet, il n'est pas possible

que les deux valeurs reçues à une porte de multiplication dépendent des y, pour obtenir A_i à partir de A il suffit d'annuler les y_j, $j \neq i$, et de supprimer les additions d'une porte dépendant de y_i avec une porte qui n'en dépend pas. Ces circuits relèvent de l'hypothèse de récurrence, et calculent les coefficients $a_i(\underline{x})$ quand on y remplace y_i par 1. Quant au coefficient $a_0(\underline{x})$, il se calcule par un circuit A_0 extrait de C, obtenu en annulant tous les y, qui relève aussi de l'hypothèse de récurrence. Ces parallélisations n'utilisent que la constante 1 comme paramètre, car la constante 0 ne sert à rien dans un calcul de polynôme : on peut toujours en éliminer les additions à 0 et les multiplications par 0.

Tous ces circuits se parallélisent donc en profondeur $m.\log(8c)$; il faut ensuite remplacer les y_i par le produit de valeurs calculées dans le parallélisé de B, les multiplier par le a_i correspondant et faire la somme, ce qui demande une profondeur supplémentaire de $2 + \log(c) + 1 = \log(8c)$.

Le sous-degré est conservé, mais pas le degré car on a remplacé des variables par 1 ; une récurrence immédiate montre qu'il reste inférieur à 3^m. L'ampleur est automatiquement conservée car on n'emploie que la constante 1.

Comme nous sommes en profondeur polylogarithmique, si nous calculons à chaque étape les a_i isolément, nous dépassons la complexité polynomiale. Il nous faut coordonner leur calcul. Pour cela, nous introduisons la notion de dérivée $\partial p/\partial q(\underline{x},\underline{y})$ du polynôme calculé à la porte p d'un circuit par rapport à une autre porte q : il s'agit de la dérivée par rapport à la variable y du polynôme $Q_p(\underline{x},y)$ calculé en p dans le circuit obtenu à partir de C en remplaçant la porte q par une entrée étiquetée par une nouvelle variable y ; en fait, nous n'avons besoin de calculer cette dérivée que lorsque Q est de degré un en y, si bien que nous convenons que ce polynôme est nul si Q ne dépend pas de y, ou bien s'il est de degré au moins deux en y. Dans les autres cas, il est défini par l'induction suivante : $\partial p/\partial p(\underline{x},\underline{y}) = 1$; si $p = p' + p''$, $\partial p/\partial q(\underline{x},\underline{y}) = \partial p'/\partial q(\underline{x},\underline{y}) + \partial p''/\partial q(\underline{x},\underline{y})$; si $p = p'.p''$, où $Q_{p'}$ dépend de y (à la différence de $Q_{p''}$), $\partial p/\partial q(\underline{x},\underline{y}) = Q_{p''}.\partial p'/\partial q(\underline{x},\underline{y})$.

Nous considérons un multicircuit C, sans constantes, de taille c, de degré $d \leq 2^m$, et le multicircuit C^*, de même degré et de taille c^2 qui calcule non seulement le polynôme arrivant à chaque porte de C, mais aussi ses dérivées par rapport aux autres portes ; aux sorties de C^*, on obtient les polynômes calculés par C, et les dérivées des sorties de C par rapport à n'importe quelle porte de C ; il nous suffit de montrer qu'on peut paralléliser C^* en profondeur $(m+1).\log(8c)$ et en complexité $2.3^m.c^3$; pour la complexité, nous considérons la suite u_m définie par $u_0 = 1/c$ et la relation de récurrence $u_{m+1} = 3.u_m + 3$, et nous montrons par récurrence sur m que la complexité de la parallélisation est inférieure à $u_m.c^3$; c'est suffisant puisque $u_m < (3/2 + 1/c).3^m \leq 2.c^3$, hormis le cas trivial où $c = 1$.

Si $m = 0$, il n'y a que des additions, C^* se parallélise en profondeur $\log(c) + 1 = \log(2c)$ et en taille $c^2 \leq c^3$ (les dérivées sont des entiers, qu'on calcule par additions de 1).

Si $m \geq 0$, on considère les portes p_1 , ... p_k de C définies comme ci-dessus ; les multicircuits B^* , A^* et A_0^* relèvent de l'hypothèse de récurrence. Si le polynôme calculé à une sortie s de C^* est $a_1(\underline{x}).y_1 + ... + a_k(\underline{x}).y_k + a_0(\underline{x})$, les coefficients $a_i(\underline{x})$ sont calculés comme dérivées dans A^* , et $a_0(\underline{x})$ est calculé à une sortie s_0 de A_0^* ; pour dériver ces polynômes par rapport à une porte q de C , on calcule $\partial s/\partial q = a_1.\partial p_1/\partial q + ... + a_n.\partial p_n/\partial q + \partial s_0/\partial q$ en branchant le calcul des $\partial p_i/\partial q$ sur B^* . La profondeur est la même qu'auparavant ; pour la complexité, les parallélisations de A^* , de A_0^* et de B^* ne demandent chacune pas plus de $u_m.c^3$ opérations ; il y a au plus c^2 sorties dans C^* , chacune ne demandant pour être calculée à partir de A^* , A_0^* et B^* que $3c$ opérations de plus ; la complexité totale est donc $3.u_m.c^3 + 3.c^3 = u_{m+1}.c^3$. **Fin**

On obtient également une variante du théorème précédent, basé sur l'ampleur et non pas sur le degré, qui est un peu meilleure quand $\log(a) < p$.

Théorème 7.3. *Un circuit* C , *de complexité* $c > 0$, *d'ampleur* $a > 1$ *et de degré* d , *se remplace par un circuit équivalent, de profondeur* $\log(\log(a)).\log(8c) + \log(8d)$, *de complexité* $6.\log(a)^{\log(3)}.c^3$, *de degré* $d.\log(a)^{\log(3)}$ *et de même ampleur, qui calcule le même polynôme, et qui utilise la constante* 1 *en plus des constantes de* C .

Démonstration. Nous considérons un multicircuit sans constantes M , de complexité c et de profondeur p , le maximum des ampleurs de ses sorties étant a . Si a est inférieur ou égal à 4 , ses sorties calculent des polynômes qui ne comprennent pas plus de 4 monômes : chaque monôme se calcule en profondeur au plus $\log(d) + 1$, et la profondeur deux suffit pour les sommer ; on obtient alors une profondeur $\log(d) + 3 = \log(8d)$ inférieure à celle annoncée.

Pour $a \geq 4$, nous procédons par induction sur a ; nous notons p_1 , ... p_k les portes minimales dans C d'ampleur strictement supérieure à $\sqrt{a}$; ces portes ne sont pas des entrées, et nous notons p_i' et p_i'' les portes qui envoient leurs flèches vers p_i .

Soit B le sous-multicircuit de C dont les sorties sont les p' et les p'' ; ces dernières portes ont une complexité inférieure à c , et une ampleur inférieure ou égale à $\sqrt{a}$, si bien que l'hypothèse d'induction s'applique, ou bien ce nombre est inférieur à 4 ; dans tous les cas, on peut paralléliser B en profondeur $\log(\log(\sqrt{a})).\log(8c) + \log(8d) = (\log(\log(a)) - 1).\log(8c) + \log(8d)$.

Notons A le circuit obtenu en remplaçant p_1 , ... p_k par des entrées indexées par des variables y_1 , ... y_k , en supprimant pour chacune d'entre elles les deux flèches qu'elle reçoit, et en éliminant les portes qui ne sont plus sur un chemin arrivant à la sortie de C . Comme chacune des portes p_i est d'ampleur au moins $\sqrt{a}$, les sortie de A calcule un polynôme au plus du premier degré en y_1 , ... y_k , de la forme $a_1(\underline{x}).y_1 + ... + a_k(\underline{x}).y_k + a_0(\underline{x})$. Les polynômes $a_1(\underline{x})$, ... $a_k(\underline{x})$ se calculent par des circuits A_1 , ... A_k qui relèvent de l'hypothèse de récurrence ; on remarquera qu'ils utilisent la constante 1 . Quant au coefficient $a_0(\underline{x})$, il se calcule par un circuit A_0 extrait de C qui relève aussi de l'hypothèse de récurrence.

Tous ces circuits se parallélisent donc en profondeur $(\log(\log(a)) - 1).\log(8c) + \log(8d)$; il faut ensuite remplacer les y_i par leurs valeurs calculées dans le parallélisé de B , les multiplier par le a_i correspondant et faire la somme, ce qui demande une profondeur supplémentaire de $\log(c) + 3 = \log(8c)$; en l'additionnant à la précédente on obtient $\log(\log(a)).\log(8c) + \log(8d)$.

Par hypotèse d'induction, les parallélisés de B et des A_i sont de degré inférieur à $d.\log(\sqrt{a})^{\log(3)}$, si bien que le degré du parallélisé de C est borné par $3.d.\log(\sqrt{a})^{\log(3)} = d.\log(a)^{\log(3)}$. L'ampleur est automatiquement borné par a du fait que seule la constante 1 est utilisée.

Pour borner la complexité, on procède comme dans le cas du degré, en supposant que $\log(a) \leq 2^m$, on montre que la parallélisation se fait en complexité $u_m.c^3 < 2.3^m.c^3$, où la suite u_m est définie par $u_0 = 1/c$ et la relation de récurrence $u_{m+1} = 3.u_m + 3$.

La seule différence est qu'il faut amorcer la récurrence pour $a \leq 4$, c'est-à-dire pour $m = 1$; la complexité est c s'il n'y a que des multiplications ; sinon, chaque monôme ne demande pas plus de $c-1$ multiplications, et il y en a pas plus de quatre à sommer à chacune des sorties, d'où une complexité du parallélisé inférieure à $c.(4(c-1) + 3) \leq 4.c^2 \leq u_2.c^3 = (3 + 1/c).c^3$. **Fin**

8. Questions de degrés

Si un polynôme est donné par un circuit utilisant des constantes, on ne connaît pas de méthode rapide qui permette de trouver son véritable degré. Par contre, si un polynôme est calculé par un circuit de complexité c , et s'il se trouve qu'on sait que son degré est majoré par d , on pourra grâce au Lemme d'homogénéisation le calculer par un circuit, de taille raisonnable, dont on voit par une simple inspection de sa forme qu'il ne peut calculer un polynôme de degré supérieur à d .

C'est pour cela que nous avons majoré le degré réel, en quelque sorte accidentel, du polynôme calculé par un degré formel lisible dans le circuit. Il est évident qu'un circuit de sous-degré d calcule un polynôme de degré inférieur ou égal à d. Réciproquement, si un circuit C de complexité c calcule un polynôme f qui se trouve être de degré d, f est égal à sa troncature au degré d, qui, nous l'avons vu, se calcule par un circuit C' de complexité $c.(d+1)^2$, et de sous-degré d.

Fort bien. Mais cette notion de degré formel traite différemment les variables et les constantes, ce qui peut nous gêner dans un contexte où l'on doit substituer des constantes aux variables : on perd la trace du degré du polynôme dans lequel on a effectué la substitution. C'est pour cela que nous avons défini le degré d'un circuit de la même façon que son sous-degré, sauf que, cette fois, les constantes sont elles-aussi de degré un.

Lemme 8.1 (ELIMINATION DES CONSTANTES, [MALOD 2003]). *Un circuit* C *de complexité* c *et de sous-degré* d, *peut être remplacé par un circuit de complexité inférieure et de sous-degré* $c.d + 1$, *qui utilise comme constantes celles qui sont calculées aux portes de degré formel nul de* C, *et qui calcule le même polynôme.*

Démonstration. Nous faisons subir à C la manipulation suivante : si une porte de sous-degré n'est pas une entrée, on supprime les deux flêches qui y arrivent, si bien qu'on la transforme en une entrée, qu'on étiquette par la constante qui y était calculée ; on obtient ainsi un multicircuit Γ, dont on jette ensuite toutes les portes qui n'interviennent pas dans le calcul de la sortie. Le circuit Γ' ainsi obtenu est de complexité inférieure à c, et calcule le même polynôme que C.

Montrons par induction sur la complexité c de l'ancien circuit que le degré de Γ' satisfait la majoration indiquée. C'est évident si $c = 0$, ou bien si la sortie p de C est de sous-degré nul. Sinon, elle reçoit dans les deux circuits ses flêches de portes p' et p''. Notons d, d' et d'' les sous-degrés de ces portes dans C, et δ, δ', δ'' leur degrés dans Γ'. Par hypothèse d'induction $\delta' \leq (c-1).d' + 1$ et $\delta'' \leq (c-1).d'' + 1$. Si p est une porte d'addition, $\delta = \max(\delta',\delta'') \leq (c-1).\max(d',d'') + 1 \leq c.\max(d',d'') + 1 = c.d + 1$. Si p est une porte de multiplication, $\delta = \delta' + \delta'' \leq (c-1).(d' + d'') + 2 = (c-1).d + 2 \leq c.d + 1$ puisque $d \geq 1$. **Fin**

Remarquons que, si on multiplie l'inconnue x successivement par les constantes $c_1, \ldots c_d$, on obtient un circuit de degré $d+1$.

Il faut bien noter que, contrairement aux lemmes précédents où on remplace des circuits par des circuits équivalents, les constantes qui sont utilisées cette fois dans le nouveau circuit ne sont pas les mêmes que celles de l'ancien circuit, mais que ce sont des constantes calculées à partir de ces dernières : c'est inévitable, car borner le degré bride le calcul des constantes, ce que ne fait pas une simple borne du sous-degré.

Plutôt que de chercher à minimiser deux paramètres, la complexité et le degré, il est avantageux de pouvoir n'en considérer qu'un seul, la complexité, en nous limitant à une classe particulière de circuits, caractérisée par un critère formel, intermédiaire entre les termes et et les circuits les plus généraux. C'est ce que permet le Lemme de Malod ci-dessous.

Nous avons vu que ce qui caractérise les termes, c'est que les deux sous-circuits d'où une porte d'addition ou de multiplication reçoit ses flêches sont disjoints. Nous dirons qu'un circuit est multiplicativement disjoint s'il a cette propriété pour les portes de multiplication : si une porte p de multiplication reçoit ses flêches de p' et de p'', le circuit $C(p')$ formé des portes situées sur un chemin joignant une entrée à p' est disjoint du circuit $C(p'')$ formé des portes situées sur un chemin joignant une entrée à p''.

Lemme 8.2 (CONTRAINTES SUR UN POLYNOME CALCULE PAR UN CIRCUIT MULTIPLICATIVEMENT DISJOINT). *Un circuit multiplicativement disjoint de complexité* c *est de degré inférieur à* $c + 1$ *; le nombre de monômes du polynome qu'il calcule est majoré par* 2^c *; s'il n'utilise que la constante* -1 *, il calcule un polynôme à coefficients entiers de valeurs absolues majorées par* 2^c.

Démonstration. Montrons d'abord la majoration du degré. C'est clair pour une entrée ; pour une porte d'addition, le degré est majoré par $\max(c'+1,c''+1) \leq c$, où c' et c'' désigne les complexités des deux sous-circuits arrivant à la porte en question ; si c'est une porte de multiplication, son degré est majoré par $(c'+1) + (c''+1) = c + 1$ puisque, vu la disjonction des sous-circuits, $c = c' + c'' + 1$.

Pour les monômes, et le module des coefficients, on majore l'amplitude du circuit ; elle vaut 1 aux entrées ; pour une addition, elle est majorée par $2^{c'} + 2^{c''} \leq 2.2^{c-1} = 2^c$, puisque $c', c'' \leq c-1$; pour une multiplication, elle est inférieure à $2^{c'}.2^{c''} < 2^c$. **Fin**

On remarque que nous obtenons pour les circuits multiplicativement disjoints exactement les mêmes contraintes que pour les termes. Et, en effet, on ne sait pas s'il est possible de transformer sans explosion de complexité un circuit multiplicativement disjoint en un terme ; on doit se contenter de penser que c'est peu probable.

Lemme 8.3 (DE MALOD, [MALOD 2003]). *Un circuit de complexité* c *, et de degré* d *, se remplace par un circuit multiplicativement disjoint de complexité* $c.d$ *, qui calcule, avec les mêmes constantes, le même polynôme.*

Démonstration. Nous construisons un circuit C' comprenant, pour chaque porte p de C, dont nous notons $d(p)$ le degré, $d + 1 - d(p)$ copies $p_0, \ldots p_{d-d(p)}$ de cette porte, qu'on appellera les clones de p ; pour chacun d'entre eux, les circuits $C(p)$ et $C'(p_i)$ sont isomorphes, si bien qu'ils calculent tous le même polynôme

que p . En particulier, la sortie de C' , qui est l'unique clône de celle de C , calcule le même polynôme que cette dernière.

Nous procédons ainsi : si p est une porte d'addition, somme des portes q et q' , son clone p_k est la somme du k° clone q_k de q et du k° clone q'_k de q' , qui existent bien puisque ces portes, étant de degré inférieur, ont au moins autant de clones que p . Si p est une porte de multiplication, qui reçoit ses flèches de q et de q' , son clone p_k , $0 \leq k \leq d - d(p) = d - d(q) - d(q')$, est une porte de multiplication qui reçoit ses flèches du clone q_k et du clone $q'_{k+d(q)}$, qui existent bien puisque $0 \leq k \leq d - d(q)$ et $0 \leq k + d(q) \leq d - d(q')$.

On observe, grâce à une facile induction sur la profondeur que les portes du sous-circuit $C'(p_k)$ associé au k° clone de p sont des clones d'indices compris au sens large entre k et $k + d(p) - 1$. Cette condition garantit que le circuit C' est multiplicativement disjoint. Comme chaque porte n'a pas plus de d clones, sa complexité est bien bornée par $c.d$. **Fin**

Note. La démonstration ci-dessus est due à Natacha Portier. On peut observer que ce lemme n'utilise pas de propriétés particulières de la multiplication, sauf que le degré formel est en rapport avec ce qu'il est usuel d'appeler le degré d'un polynôme. C'est ainsi que nous pouvons définir le degré termique d'un circuit en traitant les portes d'addition comme les portes de multiplication : il s'agit du nombre calculé en mettant des 1 aux extrémités du circuit et en remplaçant toutes les autres portes par des additions. On voit sans peine qu'un terme de complexité c est de degré termique $c+1$, tandis qu'un circuit de degré termique t se remplace par un terme de complexité $t-1$. Le Lemme de Malod est un résultat de ce genre, où on compte le degré seulement pour un sous-ensemble des portes ; la complexité intervient dans la taille du circuit partiellement disjoint, car on ne peut négliger complètement les autres portes.

9. Un vocabulaire d'usage courant : les classes de complexité

Le lecteur aura compris que nous considérons comme impraticable un calcul demandant un nombre exponentiel d'opérations ; nous adoptons à partir de maintenant la convention qu'un calcul faisable est un calcul qui ne demande qu'un nombre polynomial d'opérations.

On peut exprimer les théorèmes et les conjectures de la théorie des calculs faisables en termes de transformations de circuits, avec controle de leurs paramètres, comme nous l'avons fait jusqu'à présent. Mais il est plus courant de le faire en termes d'égalité ou d'inégalité de classes de complexité, si bien que l'auteur de ces lignes se voit contraint de définir ces classes, bien qu'il préfère la première façon.

Dire que le calcul d'un polynôme donné, par exemple le déterminant d'ordre 100 , est de complexité polynomiale n'a aucun sens ; par contre, il est signifiant de dire que, pour tout n , le calcul de Det_n se fait en n^5 opérations. Pour parler de complexité polynomiale, ce qu'il nous faut, c'est introduire des suites f_n de polynômes et mesurer le comportement assymptotique de la suite c_n de leur complexité : c'est par abus de langage que nous disons que "le déterminant" est calculable en temps polynomial ; nous parlons en réalité d'une propriété de la suite Det_n . Dire que la suite des complexités c_n est polynomialement bornée, cela veut dire qu'il existe un polynôme $p(x)$ dans $R[x]$ tel que, pour tout n , $c_n \leq p(n)$; cela veut dire encore qu'il existe deux constantes positives A et B telles que, pour tout n , $c_n \leq n^A + B$.

Si K est un anneau (commutatif et intègre), et A une partie de K , nous dirons qu'une suite f_n de polynômes en plusieurs variables, à coefficients dans K , est dans la classe $VPdl(A;K)$ s'il existe un polynôme $p(x)$ tel que, pour tout n , f_n soit calculé par un circuit de complexité inférieure à $p(n)$, dont les entrées sont étiquetées par des variables et des constantes prises dans A .

Quand $A = \{a_1, \ldots a_k\}$ est fini, nous écrivons $VPdl(a_1, \ldots a_k;K)$ au lieu de $VPdl(\{a_1, \ldots a_k\};K)$. Si l'anneau K est clairement indiqué par le contexte, nous l'omettons, et en particulier nous écrivons $VPdl(K)$ et non pas $VPdl(K;K)$. C'est ainsi que $VPdl(0)$, $VPdl(1)$, $VPdl(-1)$ désignent des classes de suites de polynômes à coefficients entiers. On notera $VPdl(\text{mod } p)$ plutôt que $VPdl(Z/pZ)$.

La constante 0 ne sert a rien dans le calcul d'un polynôme non-nul ; on peut en effet l'éliminer des circuits en supprimant les portes d'addition à 0 , et en remplaçant par une entrée nulle les portes de multiplication par 0 . Pour cette raison, nous préferons souvent la notation $VPdl(0)$ à la notation $VPdl(\varnothing)$. Nous considérerons à l'occasion la classe $VPdl(-0)$, intermédiaire entre $VPdl(0)$ et $VPdl(-1)$, où la constante -1 n'est utilisée que pour des changements de signe : les seules portes de sous-degré nul sont des entrées étiquetées par -1 , qui n'ont le droit d'envoyer leurs flêches que vers des portes de multiplication ; toutes les autres portes calculent un polynôme sans terme constant.

On voit qu'une suite de $VPdl(A)$ a un nombre polynomial en n de variables, tandis que son degré et son nombre de monômes sont (simplement) exponentiels en n . S'il s'agit de $VPdl(-1)$, les coefficients de ces monômes sont doublement exponentiels en n .

On introduit également des suites de polynômes de plus en plus strictes où non seulement la complexité, mais aussi le degré, sont polynomialement bornés.

Nous dirons qu'une suite f_n de polynômes en plusieurs variables, à coefficients dans K, est dans la classe $VPdb(A;K)$ s'il existe un polynôme $p(x)$ tel que (i) pour tout n, le degré total de f_n est majoré par $p(n)$ (ii) pour tout n, f_n est calculé par un circuit de complexité inférieure à $p(n)$, dont les entrées sont étiquetées par des variables et des constantes prises dans A. D'après le lemme d'homogénéisation, il est équivalent de demander que f_n soit calculé par un circuit, avec paramètres dans A, dont la complexité et le sous-degré soient polynomialement bornés.

Le degré est polynomial dans cette classe, mais comme le calcul des constantes n'y est pas bridé par cette borne du degré, les coefficients restent doublement exponentiels dans $VPdb(-1)$.

Nous dirons ensuite qu'une suite f_n de polynômes en plusieurs variables, à coefficients dans K, est dans la classe $VPmd(A;K)$ s'il existe un polynôme $p(x)$ tel que pour tout n, f_n soit calculé par un circuit multiplicativement disjoint de complexité inférieure à $p(n)$, dont les entrées sont étiquetées par des variables et des constantes prises dans A. Il est équivalent de borner polynomialement la complexité et le degré de la suite de circuits calculant f_n.

Cette fois, les coefficients sont simplement exponentiels dans $VPmd(-1)$.

Nous dirons enfin qu'une suite f_n de polynômes en plusieurs variables, à coefficients dans K, est dans la classe $VPt(A;K)$ s'il existe un polynôme $p(x)$ tel que pour tout n, f_n soit calculé par un terme de complexité inférieure à $p(n)$, dont les entrées sont étiquetées par des variables et des constantes prises dans A.

Examinons tout d'abord quelques rapports évidents entre ces classes :

(i) Pour chaque A, $VPt(A)$ est inclus dans $VPmd(A)$, qui est inclus dans $VPdb(A)$, lui-même inclus dans $VPdl(A)$; pour $A = \{-1\}$, les deux dernières inclusions sont strictes, mais on ne sait ce qu'il en est de la première ; on ne sait même pas montrer que le déterminant n'est pas dans $VPt(-1)$!

(ii) Si A est inclus dans B, $VPt(A)$, $VPmd(A)$, $VPdb(A)$ et $VPdl(A)$ sont inclus respectivement dans $VPt(B)$, $VPmd(B)$, $VPdb(B)$ et $VPdl(B)$.

(iii) Si B est fini et inclus dans le demi-anneau engendré par A, $VPt(B)$, $VPmd(B)$, $VPdb(B)$ et $VPdl(B)$ sont inclus respectivement dans $VPt(A)$, $VPmd(A)$, $VPdb(A)$ et $VPdl(A)$; en effet, le calcul des éléments de B à partir de A n'augmente que d'une constante la complexité des circuits. C'est ainsi que $VPdl(-1)$ contient $VPdl(1)$. Bien que Z et -1 engendrent le meme anneau, $VPdl(Z)$ contient strictement $VPdl(-1)$: dans la première classe on peut employer

gratuitement des constantes entières arbitrairement grandes, tandis que dans la seconde on prend en compte le temps mis à les calculer.

(iv) De même, si B s'obtient en ajoutant à A un nombre fini de constantes calculables à partir de A, $VPdl(A) = VPdl(B)$, $VPdb(A) = VPdb(B)$, $VPmd(A) = VPmd(B)$, $VPt(A) = VPt(B)$.

(v) Si K est un anneau, d'après le lemme d'élimination des constantes $VPmd(K) = VPdb(K)$. En particulier $VPmd(\text{mod } p) = VPdb(\text{mod } p)$. De même $VPmd(0) = VPdb(0)$. Par contre $VPdb(-1)$ contient strictement $VPmd(-1)$, au vu de la taille des coefficients.

(vi) Quand on remplace les constantes par des variables dans une suite de $VPt(A)$, $VPmd(A)$ ou $VPdl(A)$, on obtient une suite respectivement dans $VPt(0)$, $VPmd(0)$ ou $VPdl(0)$. Autrement dit les classes $VPt(A)$, $VPmd(A)$ et $VPdl(A)$ s'obtiennent par substitutions de constantes prises dans A à des variables à partir de $VPt(0)$, $VPmd(0)$ et $VPdl(0)$. Cela n'est plus vrai pour $VPdb$, ce qui fait de cette dernière une classe un peu bâtarde.

Nous avons pu calculer un déterminant d'ordre n en $O(n^5)$ opérations, en utilisant seulement la constante -1, ce qui met le déterminant dans la classe $VPdl(-1)$, et bien sûr dans $VPdb(-1)$ puisqu'il est de degré n ; cela le met automatiquement dans $VPmd(Z)$. Mais on peut en outre faire l'observation suivante : quand on calcule le déterminant des $1+y_{ii}$ et des y_{ij}, avec division finale simulée, et qu'on remplace toutes les inconnues par 0, on calcule le determinant de la matrice-identité, qui n'est jamais modifiée par le pivot de Gauss, et à la fin on divise (ou plutôt, on multiplie !) 1 par 1 ; en conséquence, tous les polynômes qui apparaissent au cours du calcul de ce déterminant ont pour terme constant 0, 1 ou -1 ; il en est de même dans la troncature du circuit au degré n. Autrement dit, dans ce calcul du déterminant les portes de sous-degré nul calculent seulement 0, 1 ou -1, et il en est de même pour le déterminant obtenu après remplacement de y_{ii} par $x_{ii}-1$. D'après le lemme d'élimination des constantes, on obtient un circuit de degré formel complet $O(n^6)$ qui n'utilise que les constantes 1 et -1, puisque 0 s'élimine ; le remplacement de 1 par sa valeur $1 = (-1)^2$ ne fait que doubler le degré. En conclusion, le déterminant est dans $VPmd(-1)$.

On ne sait pas si le déterminant est dans $VPt(-1)$, les meilleures parallélisations du calcul du déterminant actuellement connues se faisant en profondeur polylogarithmique, et dans la classe $VPmd(-0)$ (voir [MV 1999]) ; la méthode de Gauss a une très mauvaise profondeur $O(n.\log(n))$. Comme nous l'avons déja dit, on ne sait pas non plus si la classe $VPt(-1)$ est strictement incluse dans la classe $VPmd(-1)$: c'est vraisemblablement un problème très difficile. Il est utile de lui donner une autre formulation :

QUESTION VPmd(-1) =? VPt(-1) . *Existe-t'il une constante* A *telle que tout circuit arithmétique multiplicativement disjoint de complexité* c *, n'utilisant que la constante* -1 *, se remplace par un terme de complexité inférieure à* c^A *, utilisant la même constante, et qui calcule le même polynôme ?*

Démonstration. C'est la seule démonstration de ce type que nous expliciterons, les autres étant laissées au lecteur. Il est clair que si cette constante existe VPmd(-1) = VPt(-1) . Si elle n'existe pas, on peut pour chaque n trouver un polynôme f_n calculé par un circuit mutiplicativement disjoint de complexité c_n et pas par un terme de complexité c_n^n ; comme f_n est calculé par un terme de complexité $2^{c_n} - 1 \leq c_n^{c_n}$, $n < c_n$, si bien que la suite des c_n n'est pas bornée ; on peut la remplacer par une suite extraite strictement croissante qui a la même propriété.

Si donc la suite des c_n est croissante, on considère la suite g_m de polynômes qui vaut f_n si $m = c_n$, et 0 si m n'est pas de la forme c_n . Cette suite est dans VPmd(-1) et pas dans VPt(-1) . **Fin**

Un autre problème largement ouvert, c'est de savoir si le permanent est dans VPdb(-1) . La conjecture est qu'il n'y est pas. On sait seulement qu'il n'est pas possible de le calculer rapidement sans faire des soustractions, c'est-à-dire qu'il n'est pas dans VPdb(N) (voir [SV 1998]).

Dans ces notations, la lettre V rend hommage à Leslie Valiant, qui a introduit ce type de calcul ; P est pour polynomial, dl pour "degré libre", db pour "degré borné", md pour "multiplicativement disjoint" et t pour "terme".

La classe originelle considérée par Valiant est VPdb(K) . Comme on peut penser qu'il est trop généreux, et très peu réaliste du point de vue algorithmique, de pouvoir utiliser librement des constantes arbitraires prises dans un corps K (il n'est même pas exigé que ce soit le même ensemble fini de constantes qui soit utilisé dans le calcul de chaque f_n), il est naturel d'introduire la classe VPdb(-1) , et surtout sa compagne plus stricte VPmd(-1) . Il est aussi naturel de relâcher la contrainte sur le degré, ne serait-ce pour mieux comprendre comment elle intervient dans la problématique de Valiant. Nous avons d'ailleurs été témoins de son rôle essentiel dans l'élimination des divisions du calcul des déterminants. Toutes ces variantes des classes VP ont été introduites par Guillaume Malod dans sa thèse, soutenue en 2003. Nous avons modifié les notations de Malod, qui emploie VPnb pour VPdl , VP pour VPdb , VP° pour VPmd et VPe pour VPt .

Par contre, dans nos définitions, nous ne poussons pas le réalisme algorithmique jusqu'à rendre compte de l'uniformité du calcul. Expliquons-nous : pour calculer des déterminants d'ordre n , nous utilisons la "même" méthode que pour calculer un déterminant d'ordre n' ; c'est bien pour cela que nous pouvons affirmer en toute certitude que nous savons calculer "le" déterminant. On peut définir ce qu'on appelle des classes uniformes, en formalisant une notion de similitude de méthode de calcul pour tous les termes de la suite f_n . Avec les conventions que nous adoptons ici, la meilleure méthode pour calculer un déterminant d'ordre n n'a peut-être rien à voir avec la meilleure méthode pour calculer un déterminant d'ordre $n+1$; on peut aussi penser que le meilleur calcul d'un déterminant

numérique n'a rien à voir avec le calcul du déterminant comme polynôme, ni même avec les propriétés algébriques ou combinatoires du déterminant ! Il est plus que douteux que cette philosophie mène à quoi que ce soit de sensé, et ce n'est pas la raison pour laquelle nous négligeons l'uniformité : c'est plutôt parce que chacun des théorèmes que nous montrons peut s'accompagner d'une version uniforme, dont la démonstration consiste à faire la constatation évidente, quoique parfois laborieuse, que nos manipulations de circuits se font uniformément. En d'autres termes, l'uniformité ne serait pour nous qu'un ajout parasite, qui n'éclairerait en rien notre problématique.

On ne perdra pas de vue qu'un modèle de calcul, quel qu'il soit, n'est qu'une idéalisation de la notion de calcul véritable (par exemple, nous considérons comme réaliste un calcul qui se fait en temps polynomial, alors qu'un informaticien bien né réfléchira à deux fois avant d'entreprendre un calcul en temps quadratique), ce qui ne signifie pas qu'il soit sans intérêt pour la description du monde réel. En fait, l'expérience, à défaut d'autre chose, prouve que notre modèle rend compte fidèlement de phénomènes constatés dans le monde réel. Il convient aussi, quand on parle d'un modèle de calcul, de ne jamais oublier les conventions sur lesquelles il repose.

SUMMARY OF THE NINE FIRST SECTIONS

VPxx(A) is the class of sequences of polynomials, submitted to various algorithmic constraints depending on the letters xx ; A denotes the constant parameters that we may use in course of computation ; the ring K generated by A is understood from the context : the polynomials in VPxx(0) , VPxx(1) , VPxx(-1) have integer coefficients ; in case of necessity, we may write VPxx(A ; K) .

VPdl(A) is the class of sequences of polynomials computed by a sequence of circuits of polynomially bounded complexity.

> Their number of variables is polynomially bounded ; their degree and their number of monomials are simply exponential ; if A = {-1}, the coefficients of their monomials are doubly exponential integers.

VPdb(A) is the class of sequences of polynomials of polynomially bounded degree, which are computed by a sequence of circuits of polynomially bounded complexity ⇔ they are computed by a sequence of circuits of complexity and sub-degree polynomially bounded.

> Their number of variables and their degree are polynomially bounded ; their number of monomials is simply exponential ; if A = {-1}, the coefficients of their monomials are doubly exponential integers.

VPmd(A) is the class of sequences of polynomials computed by a sequence of multiplicatively disjoints circuits of polynomially bounded complexity ⇔ they are computed by a sequence of circuits of complexity and degree polynomially bounded.

Their number of variables and their degree are polynomially bounded ; their number of monomials is simply exponential ; if A = {-1}, the coefficients of their monomials are simply exponential integers. If A is closed under addition and multiplication, VPdb(A) = VPmd(A) .

VPt(A) is the class of sequences of polynomials computed by a sequence of terms of polynomially bounded complexity ⇔ they are computed by a sequence of circuits of logarithmic depth.

Their number of variables and their degree are polynomially bounded ; their number of monomials is simply exponential ; if A = {-1}, the coefficients of their monomials are simply exponential integers.

The determinant is in VPmd(-1) and the permanent is not in VPdb(N) ; it is not known wether the first is in VPt(-1) nor the second in VPdb(-1) . It is not known wether the inclusion of VPt(-1) in VPmd(-1) is proper.

10. Sommes de Valiant.

Nous avons dit qu'un polynôme est une somme de monômes ; nous avons même observé que c'est une somme simplement exponentielle de monômes ; cela rend naturel la définition d'une autre opération sur les polynômes, la sommation de Valiant.

Etant donné un polynôme f , nous séparons ses variables en deux groupes $\underline{x}$ et $\underline{y}$ et nous considérons le polynôme $g(\underline{x}) = \Sigma_{\underline{y} \in \{0,1\}^m} f(\underline{x},\underline{y})$; dans cette somme, on donne toutes les valeurs possibles, prises dans l'ensemble {0, 1} , à toutes les coordonnées du m-uple de variables $\underline{y}$, si bien qu'elle comporte 2^m termes, où m désigne la longueur du uple $\underline{y}$. Cette somme faisant intervenir un nombre exponentiel de termes, on obtient immédiatement la borne exponentielle $2^{c+1}.(c+1)$ pour la complexité de g , en fonction de la complexité c de f . Il est peu vraisemblable qu'on puisse borner polynomialement la complexité de g en fonction de celle de f , mais on ne sait pas le démontrer : c'est ça l'analogue de la question P = ? NP (et nous verrons dans la section 12 un rapport entre ces deux questions).

Si l'ensemble $\underline{y}$ de variables est vide, on convient que la somme de Valiant est sans effet, c'est-a-dire que $g(\underline{x}) = f(\underline{x})$.

Notons que le choix des deux constantes 0 et 1 est sans importance, pourvu qu'on dispose des constantes nécessaires. En effet, $\Sigma_{\underline{y} \in \{a,b\}^m} f(\underline{x},\underline{y}) = \Sigma_{\underline{u} \in \{0,1\}^m} h(\underline{x},\underline{u})$, où h s'obtient à partir de f en remplacant y_i par $(b-a).u_i + a$.

Si g_n est une suite de polynômes telle que, pout chaque n, g_n s'obtienne par sommation de Valiant de f_n (pour un certain partage des variables de ce dernier), où la suite f_n est dans $VPdl(A;K)$, on dit que la suite g_n est dans $VSPdl(A;K)$. On définit de manière analogue les classes $VSPdb(A;K)$, $VSPmd(A;K)$ et $VSPt(A;K)$. De manière évidente, chaque classe VP est incluse dans la classe VSP correspondante, mais on ne sait pas montrer que ces inclusions sont propres, sauf dans des cas triviaux. On remarque qu'une sommation de Valiant n'affecte pas sensiblement le degré, le nombre de monômes, ni l'amplitude.

Par analogie à la classe NP, les classes VSP sont souvent notées VNP, ce qui correspond d'ailleurs à la notation originelle de Valiant ; cette lettre N signifie "non déterministe", ce qui convient très mal à ce contexte ; je lui préfère la lettre S comme sommation.

Examinons quelques rapports simples entre différentes hypothèses du type $VPxx(A) = VSPxx(A)$.

(i) Pour des raisons évidentes, $VPdl(\emptyset) \neq VSPdl(\emptyset)$, et une question du type $VP = VSP$ ne se pose que si on peut utiliser au minimum 0 et 1 comme constantes.

(ii) Si φ est un homomorphisme d'anneau, l'égalité $VPdl(A) = VSPdl(A)$ implique $VPdl(\varphi(A)) = VSPdl(\varphi(A))$; en effet, les suites de polynômes de $VPdl(\varphi(A))$ comme celles de $VSPdl(\varphi(A))$ s'obtiennent en remplaçant dans une suite de $VPdl(A)$ ou de $VSPdl(A)$ les paramètres par leurs images par φ. Par exemple, $VPdl(-1) = VSPdl(-1)$ implique $VPdl(\mathrm{mod}\ p) = VSPdl(\mathrm{mod}\ p)$ pour n'importe quel nombre premier p. La même chose vaut pour $VPdb = VSPdb$, $VPmd = VSPmd$ et $VPt = VSPt$.

(iii) Si A est inclus dans B, $VPdl(A) = VSPdl(A)$ implique $VPdl(B) = VSPdl(B)$; en effet, si dans une suite de $VSPdl(B)$ on remplace les constantes par des variables, on obtient une suite de $VSPdl(\emptyset)$, soit encore de $VSPdl(A)$: si cette dernière est dans $VPdl(A)$, on obtient bien une suite de $VPdl(B)$ en remplaçant les variables substituées par les constantes originelles. C'est ainsi que $VPdl(-1) = VSPdl(-1)$ implique $VPdl(A) = VSPdl(A)$ pour tout sous-ensemble A, permettant de calculer -1, de n'importe quel K. La même chose vaut pour $VPmd = VSPmd$ et $VPt = VSPt$, mais pas pour $VPdb = VSPdb$.

(iv) Par ailleurs, $VPdl(A) = VSPdl(A)$ implique $VPdb(A) = VSPdb(A)$; en effet, si une suite de polynôme est dans $VSPdb(A)$, cette hypothèse la met dans $VPdl(A)$, et comme son degré est polynomialement borné elle est dans $VPdb(A)$. Ce raisonnement ne vaut pas pour l'égalité $VPmd = VSPmd$, sauf bien sûr dans le cas où $VPmd = VPdb$.

(v) En l'absence de cette hypothèse hautement improbable, il n'y a pas de raison évidente pour qu'une suite dans $VSPdl(A)$ de degré polynomialement borné soit dans $VSPdb(A)$, puisque borner le degré en $\underline{x}$ dans la somme de Valiant $\Sigma_{\underline{y} \in \{0,1\}^m} f(\underline{x},\underline{y})$, n'a pas d'effet sur le degré en $\underline{y}$.

La question, ou plutôt les questions VP = ? VSP , sont liées au calcul du permanent de la manière suivante. Tout d'abord, le permanent est dans la classe VSPdb(-1) , et même dans VSPt(-1) .

Pour le voir, il nous faut tout d'abord un polynôme simple, en les n^2 variables $y_{i,j}$, qui a la propriété suivante : quand on donne à ces variables des valeurs prises dans l'ensemble {0,1} , ce polynôme vaut 1 si elles forment une matrice de permutation, c'est-à-dire s'il y a un 1 et un seul dans chaque ligne et dans chaque colonne, et 0 sinon. On prendra pour $g(\underline{y})$ le produit pour chaque i , $1 \leq i \leq n$, des $1 - y_{i,j}.y_{i,k}$ pour $1 \leq j < k \leq n$, des $1 - y_{j,i}.y_{k,i}$ pour $1 \leq j < k \leq n$ (soit encore le produit des $(y_{i,j}.y_{i,k} - 1).(y_{j,i}.y_{k,i} - 1)$ pour $1 \leq j < k \leq n$), et des $y_{i,1} + y_{i,2} + \ldots + y_{i,n}$ pour $1 \leq i \leq n$. Pour calculer g , il faut $6.n.(n(n-1)/2) + n(n-1) + n-1 < 3.n^3$ opérations, ce qui majore la complexité, et même sa complexité en tant que terme.

On remarque ensuite que le polynôme de complexité quatre $x^y = x.y + 1 - y = x.y + (-1).((-1)+y)$ vaut x si y = 1 , et 1 si y = 0 ; par conséquent le polynôme $\underline{x}\hat{}\underline{y}$, produit de tous les $x_{i,j}\hat{}y_{i,j}$, qui suffit à représenter tous les monômes du permanent, est de complexité inférieure à $4.n^2$.

Le polynôme $f(\underline{x},\underline{y}) = g(\underline{y}).(\underline{x}\hat{}\underline{y})$ est donc de complexité inférieure à $3.n^3 + 4.n^2$, et il s'agit bien d'une complexité de terme. Il est clair que le permanent s'obtient par sommation de Valiant sur f .

Par ailleurs, un célèbre théorème de Valiant affirme que le permanent possède une propriété d'universalité pour la classe VSPdb(Q) , si bien que montrer que VSPdb(Q) = VPdb(Q) revient à montrer que le permanent est dans VPdb(Q) , c'est-à-dire qu'il est calculable en temps polynomial, avec des constantes rationnelles, ce qui semble bien improbable.

Plus précisément, dans ce théorème de Valiant, seules les constantes 1 , -1 et 1/2 jouent un rôle, si bien qu'affirmer que VSPmd(-1/2) = VPmd(-1/2) revient à montrer que le permanent est dans la classe VPmd(-1/2) . Pour $p \neq 2$, c'est-à-dire si 2 est inversible modulo p , cette condition passe au quotient, et l'égalité VSPmd(mod p) = VPmd(mod p) équivaut également à la possibilité d'un calcul polynomial du permanent modulo p . Pour p = 2 , on rappelle que le permanent est facilement calculable, étant égal au déterminant. Il est cependant très peu probable que VSPmd(mod 2) = VPmd(mod 2) .

Bien que tous les coefficients du permanent soient positifs, on peut montrer qu'il n'est pas dans la classe VPmd(1) , ni même dans la classe VSPmd(1) ; autrement dit, c'est un fait établi qu'il n'y a pas de calcul efficace du permanent sans soustractions (voir [SV 1998]).

11. Savez-vous faire des additions ?

Nous considérons maintenant une version plus réaliste de la notion de calcul, où les objets manipulés sont des suites booléennes, c'est-à-dire des suites $\underline{x} = (x_1, \ldots x_n)$ prenant leur valeur dans l'ensemble $\{0,1\}$: un algorithme reçoit une suite booléenne comme donnée, et produit une suite booléenne comme résultat. Il nous faut donc mesurer la complexité de calcul de fonctions $f(\underline{x})$ de $\{0,1\}^n$ dans $\{0,1\}^m$, et principalement de $\{0,1\}^n$ dans $\{0,1\}$, que nous appelons fonctions booléennes.

On munit l'ensemble des booléens $\{0,1\}$ de la structure de corps à deux éléments $Z/2Z$. L'addition et la multiplication modulo deux sont ainsi définies : $0\oplus0 = 0$, $0\oplus1 = 1\oplus0 = 1$, $1\oplus1 = 0$, $0\otimes0 = 0\otimes1 = 1\otimes0 = 0$, $1\otimes1 = 1$. Comme dans tout corps fini, chaque fonction booléenne $f(\underline{x})$ s'exprime comme un polynôme $F(\underline{x})$; on le montre par induction sur son nombre n de variables ; c'est clair si $n = 0$, f étant la constante 0 ou la constante 1 ; pour le passage de n à $n+1$, on utilise la formule $f(\underline{x},y) = y\otimes f(\underline{x},1) \oplus (1\oplus y)\otimes f(\underline{x},0)$.

Au passage, notons l'emploi que nous venons de faire de la fonction $s(x,y,z) = x\otimes y \oplus (1\oplus x)\otimes z$; elle vaut y si $x = 1$, et z si $x = 0$, ce qui justifie son nom de sélecteur. La plupart du temps, calculer, c'est faire une suite de sélections.

Nous appellerons complexité de la fonction f le minimum de la complexité d'un polynôme F qui l'exprime, c'est-à-dire le minimum d'additions et de multiplications modulo deux qu'il faut pour obtenir f à partir des variables et des constantes 0 et 1 . Des polynômes de $(Z/2Z)[\underline{x}]$ différents peuvent donner la même fonction f , car ils n'interviennent que par leurs restrictions respectives à $Z/2Z$; comme tous les éléments de ce corps satisfont l'equation $x^2 = x$, on trouve un polynôme de degré 0 ou 1 par rapport à chacune de ses variables qui représente f ; ce polynôme est unique : en effet, si on fixe le nombre n de variables, il y a le même nombre, $2\text{^}2^n$, de fonctions et de polynômes en ces variables ; mais bien sûr ce polynôme canoniquement associé à f ne donne pas nécessairement son meilleur calcul.

Nous définissons la classe P (non uniforme ; elle est souvent notée $P/poly$) comme étant l'ensemble des suites de fonctions booléennes de complexité polynomialement bornée. Pour cette définition, nous avons fait le choix d'engendrer les fonctions à partir des opérations du corps, que nous considérons comme les seules manipulations élémentaires permises sur les booléens ; les remplacer par un nombre fini d'autres opérations, pourvu qu'elles engendrent bien toutes les fonctions, ne change pas la définition de la classe P , puisque le passage d'une base d'opération à une autre ne fait que multiplier les complexités par une constante. Il est donc assez clair que tout algorithme en temps polynomial qui manipule des booléens se traduit par des circuits arithmétiques modulo deux de complexité polynomiale.

Une autre classe intéressante est la classe Pt , habituellement notée NC_1 , des suites de fonctions décrites par des termes arithmétiques modulo deux de complexité polynomiale, soit encore des circuits arithmétiques modulo deux de profondeur logarithmique, puisque le lemme de parallélisation des termes arithmétiques est valable aussi bien en caractéristique deux qu'en caractéristique nulle. On peut d'ailleurs montrer un lemme général de parallélisation des termes, valable pour toute base d'opérations des fonctions booléennes (Théorème de Spira ; voir [POIZAT 199x] p. xx). Elle est incluse dans P , mais on ne sait pas si l'inclusion est propre, bien que l'égalité P = Pt semble très improbable ; elle signifierait que tout calcul en temps polynomial se paralléliserait en profondeur logarithmique. Elle peut être formulée de deux manières équivalentes :

QUESTION P =? Pt (version non-uniforme). *(i) Existe-t'il une constante* A *telle que tout circuit arithmétique booléen de complexité* c *puisse être remplacé par un terme arithmétique booléen de complexité* c^A *calculant la même fonction ?*
(ii) Existe-t'il une constante B *telle que tout circuit arithmétique booléen de complexité* c *puisse être remplacé par un circuit arithmétique booléen de profondeur* B.log(c) *calculant la même fonction ?*

Quand nous parlons de polynômes, l'inégalité $VP(\mathrm{mod}\ 2) \neq VPt(\mathrm{mod}\ 2)$ est évidente à cause du degré ; ce degré ne signifie plus rien pour des fonctions, qui ont plusieurs représentations polynomiales, dont une représentation canonique de petit degré (degré un par rapport à chacune des inconnues) ; une question plus modeste est la suivante :

QUESTION P =? Pmd . *(i) Existe-t'il une constante* C *telle que tout circuit arithmétique booléen de complexité* c *puisse être remplacé par un circuit arithmétique booléen multiplicativement disjoint de complexité* c^C *calculant la même fonction ?*
(ii) Existe-t'il une constante D *telle que tout circuit arithmétique booléen de complexité* c *puisse être remplacé par un circuit arithmétique booléen de complexité et de degré formel majorés par* c^D *calculant la même fonction ?*

Ces deux questions sont équivalentes car $VPdb(\mathrm{mod}\ 2) = VPmd(\mathrm{mod}\ 2)$. L'homogénéisation n'est ici d'aucun secours car si deux polynômes ont même restriction à $\{0,1\}$, il n'en est pas de même de leurs troncatures respectives ! On peut aussi poser une question un peu plus forte : est-ce bien vrai que la représentation canonique ne donne pas le meilleur calcul ?

QUESTION P =? Pc . *Existe-t'il une constante* E *telle que toute fonction booléenne de complexité* c *ait un polynôme canonique de complexité inférieure à* c^E *?*

Nous allons terminer cette section par la description de quelques algorithmes parallélisés très classiques, tout d'abord parce que c'est le petit cochon algorithmique, et qu'il faut que ce genre de techniques soit connu de nos lecteurs ; et qu'ensuite ils interviendront dans nos théorèmes.

Vous savez sans doute que tout nombre s'écrit en base dix : 2435 est formé de deux milliers, quatre centaines, trois dizaines et cinq unité. On peut choisir n'importe quelle base, et en particulier la base deux, si bien que tout nombre entier x, $0 \leq x < 2^n$, s'écrit de manière unique $\varepsilon_0 + \varepsilon_1.2 + \ldots + \varepsilon_{n-1}.2^{n-1}$, où les ε valent 0 ou 1. En effet, d'une part $\varepsilon_0 + \varepsilon_1.2 + \ldots + \varepsilon_{n-1}.2^{n-1} \leq 1 + 2 + \ldots + 2^{n-1} = 2^n - 1$; d'autre part si $(\varepsilon_0, \varepsilon_1, \ldots \varepsilon_{n-1}) \neq (\eta_0, \eta_1, \ldots \eta_{n-1})$, et si i est le plus grand indice tel que $\varepsilon_i \neq \eta_i$, avec par exemple $\varepsilon_i = 1$ et $\eta_i = 0$, $\varepsilon_0 + \varepsilon_1.2 + \ldots + \varepsilon_{n-1}.2^{n-1} - (\eta_0 + \eta_1.2 + \ldots + \eta_{n-1}.2^{n-1}) \geq 2^i - (1 + 2 + \ldots + 2^{i-1}) = 1$, si bien qu'on a ainsi une bijection entre les suites booléennes de longueur n et les nombres entiers naturels strictement inférieurs à 2^n.

Additionner deux nombres de n chiffres $(\varepsilon_0, \varepsilon_1, \ldots \varepsilon_{n-1})$ et $(\eta_0, \eta_1, \ldots \eta_{n-1})$, c'est calculer la représentation binaire $(\theta_0, \theta_1, \ldots \theta_{n-1}, \theta_n)$ de leur somme. Essayons la méthode de l'école primaire : un plus un, dix, je pose zéro et je retiens un, etc.. Si on additionne trois nombres x, y et z de un chiffre, le chiffre des unités est $u = x \oplus y \oplus z$, et le chiffre des deuzaines, c'est-à-dire la retenue, est $v = x \otimes y \oplus y \otimes z \oplus z \otimes x = x \otimes (y \oplus z) \oplus y \otimes z$; le multicircuit qui donne le résultat est de complexité cinq et de profondeur trois. Si on répète cette opération n fois, on additionne deux nombres de n chiffres (plus un nombre de un chiffre, qu'on peut prendre nul) grâce à un multicircuit de complexité $5.n$ et de profondeur $3.n$. Ce qui cause cette grande profondeur, c'est qu'on est attend de connaître la valeur de la retenue pour calculer chacun des chiffres : nous allons voir que ce n'est pas nécessaire.

LEMME 11.1 (JULES ET JIM) *On peut effectuer l'addition de deux nombres de* n *chiffres grâce à un muticircuit arithmétique booléen de profondeur inférieure à* $3.\log(n) + 3$.

Démonstration. Expliquons tout d'abord comment effectuer l'addition de deux nombres de $2n$ chiffres en répartissant le travail entre trois processeurs, que nous appelons Jules, Jim et Catherine. Cette dernière additionne les deux premières moitiés $(\varepsilon_0, \varepsilon_1, \ldots \varepsilon_{n-1})$ et $(\eta_0, \eta_1, \ldots \eta_{n-1})$ de ces deux nombres ; Jules additionne les secondes moitiés $(\varepsilon_n, \varepsilon_{n+1}, \ldots \varepsilon_{2n-1})$ et $(\eta_n, \eta_{n+1}, \ldots \eta_{2n-1})$ en faisant l'hypothèse que la retenue de Catherine sera 1, tandis que Jim fait la même chose en faisant l'hypothèse que cette retenue sera 0 : il est clair que l'un des deux travaille pour rien ! Mais, bien sûr, ils travaillent tous les deux sans attendre le résultat de Catherine, et quand c'est fini, il n'y a plus qu'à faire la choix entre les $n+1$ chiffres de Jules et ceux de Jim, en fonction de la retenue obtenue par Catherine.

Notons p_m la profondeur du circuit additionnant deux nombres de 2^m chiffres, plus un nombre de un chiffre, par itération de cette méthode ; $p_1 = 3$; $p_{m+1} = p_m + 3$, puisque la sélection se fait en profondeur trois ; donc $p_m = 3.m$. On

observe qu'après avoir additionné des nombres de un chiffre l'algorithme ne fait plus que des sélections !

Si $n \leq 2^m$, l'addition de nombres de n chiffres se ramène à celle de nombres de 2^m chiffres en ajoutant des zéros, et le plus petit m convenable est inférieur à $\log(n)+1$. **Fin**

Considérons maintenant le problème suivant : comment additionner n nombres à n chiffres ? Grâce à l'associativité de l'addition, nous pouvons les additionner deux par deux, puis additionner les résultats deux par deux, et ainsi de suite, nous ramenant ainsi à un arbre de profondeur logarithmique d'additions de deux nombres ; si nous effectuons ces additions en suivant la méthode de Jules et Jim, nous obtenons une profondeur polylogarithmique, de l'ordre de $\log^2(n)$, ce qui est beaucoup trop. Nous touchons là la difficulté majeure de la parallélisation : il faut concevoir les algoritrhmes globalement, et ne se permettre que des sous-routines de profondeur constante. D'où l'intérêt du lemme suivant :

LEMME 11.2 (ADDITIONS D'OFMAN) *Pour chaque entier* n *il existe un polycircuit arithmétique booléen de profondeur six, à* $4n$ *entrées et* $2n+2$ *sorties, qui remplace quatre nombres de* n *chiffres par deux nombres de* $n+1$ *chiffres de même somme.*

Démonstration. En profondeur trois, on remplace trois nombres par deux nombres de même somme : le premier est obtenu en additionnant les n chiffres modulo deux, et le second est celui des retenues, qui a $n+1$ chiffres car on doit lui ajouter un 0 à la place des unités. On répète l'opération avec ces deux nombres et le dernier qui reste, sans effectuer l'addition des dernières retenues. **Fin**

COROLLAIRE 11.3 (MULTIPLICATION) *La multiplication de deux nombres de* n *chifres peut s'effectuer en profondeur* $9.\log(n) + 15$.

Démonstration. Contentons-nous de perfectionner la méthode de la petite école. On doit multiplier le multiplicande par chacun des chiffres du multiplicateur, ce qui se fait en profondeur trois, grâce au sélecteur $s(x,y,0)$, et les compléter par des zéros. On doit ensuite additionner les n nombres obtenus ; on fait alors des additions d'Ofman pour diviser à chaque fois le nombres de nombres à ajouter par deux : cette étape s'effectue en profondeur $6.(\log(n)+1)$; il faut à la fin ajouter deux nombres qui n'ont pas plus de $2n$ chiffres : la méthode de Jules et Jim le fait en profondeur $3.\log(2n) + 3$.

La profondeur totale est $3 + 6.\log(n) + 6 + 3.\log(n) + 3 + 3$. **Fin**

COROLLAIRE 11.4 (DENOMBREMENT) *Pour chaque entier* n *, il existe un polycircuit arithmétique booléen, de profondeur inférieure à* $6.\log(n) + 3.\log(\log(n)) + 9$ *, à* n *entrées et à au plus* $\log(n) + 1$ *sorties, qui calcule le (développement binaire du) nombre de* 1 *contenus dans l'entrée.*

Démonstration. Le problème est d'additionner n nombres de un chiffre. Après $\log(n)+1$ étapes d'additions d'Ofman, on obtient deux nombres de $\log(n)$ chiffres qu'on somme par la méthode de Jules et Jim. **Fin**

LEMME 11.5 *Pour chaque entier* n *, il existe un circuit arithmétique booléen à* n *entrées, de profondeur inférieure à* $\log(n) + 3$ *, qui donne* 1 *à la sortie s'il y a au moins un* 1 *dans les entrées, et* 0 *s'il n'y a que des zéros.*

Démonstration. Pour obtenir un circuit de profondeur logarithmique, on pourrait itérer le dénombrement jusqu'à ce qu'il ne reste plus qu'un chiffre. Il est plus simple de procéder ainsi : on remplace chaque entrée x par son opposé $1 \oplus x$, ce qui se fait en profondeur un ; ensuite on effectue leur produit booléen $\otimes$ en profondeur $\log(n)+1$, puis on prend l'opposé de la sortie. **Fin**

12. Simulation polynomiale des calculs booléens.

La classe NP est obtenue par quantification existentielle devant la classe P . Une suite de fonctions booléennes $g_n(\underline{x})$ est dans NP s'il existe une suite $f_n(\underline{x};\underline{y})$ dans P , dont l'ensemble de variables est chaque fois partagé en deux, telle que $g_n(\underline{x}) = 1$ s'il existe un uple booléen $\underline{y}$ tel que $f_n(\underline{x};\underline{y}) = 1$, et sinon $g_n(\underline{x}) = 0$. Si on convient que $0 < 1$, $g_n(\underline{x}) = \mathrm{Max}_{\underline{y}}\, f_n(\underline{x};\underline{y})$.

Il est évident que P est inclus dans NP . On s'attend à ce qu'il y ait des suites NP de complexité irrémédiablement exponentielles, mais on ne sait pas le prouver. Autrement dit, on ne sait pas séparer la classe P de la classe NP , et la question P = ? NP est le grand problème ouvert de la Théorie de la complexité. Il y a beaucoup de raisons de penser que l'égalité de P et de NP est très peu probable, la principale étant qu'il y a de très nombreuses fonctions dans NP , associées à des problèmes naturels, pour le calcul desquelles il semble qu'il n'y ait pas de méthode efficace. C'est du moins ce qu'on espère, car l'égalité P = NP serait catastrophique pour la cryptographie, et aurait de nombreuses conséquences désastreuses sur la vie de l'homme contemporain ; par exemple, la sécurité de votre carte bleue repose sur la croyance, qui n'est pas (encore ?) fondée sur une démonstration rationnelle, que $P \neq NP$.

Lemme 12.1 (EQUIVALENCE DES TERMES ET DES CIRCUITS SOUS QUANTIFICATION EXISTENTIELLE). *A tout circuit arithmétique modulo deux* $C(\underline{x},\underline{y})$ *, de complexité* c *, est associé un terme arithmétique modulo deux* $T(\underline{x},\underline{y},\underline{z})$ *, de complexité* $4.c$ *, de profondeur inférieure à* $\log(c) + 4$ *, tel que, pour chaque* $\underline{a}$ *booléen fixé, les équations* $C(\underline{a},\underline{y}) = 1$ *et* $T(\underline{a},\underline{y},\underline{z}) = 1$ *aient exactement le même nombre de solutions booléennes. En conséquence* NP = NPt .

Démonstration. On introduit une nouvelle variable z par porte d'opération du circuit, pour représenter la valeur qui y est calculee ; l'équation $C(\underline{x},\underline{y}) = 1$

équivaut à un système de c équations de la forme $z = u \oplus v$ ou $z = u \otimes v$, et de l'équation déclarant que la variable de sortie vaut 1. A l'équation $z = u \oplus v$ on associe la fonction $z \oplus u \oplus v \oplus 1$ qui vaut 1 si elle est satisfaite, et 0 sinon ; de même, on associe la fonction $z \oplus (u \otimes v) \oplus 1$ à l'équation $z = u \otimes v$. On considère le terme $T(\underline{x},\underline{y},\underline{z})$, de complexité $4.c$, qui est le produit de tous ces termes et de la variable finale ; comme le produit est associatif, on peut effectuer toutes ces opérations en parallèle, soit en profondeur $\log(c) + 1$, à laquelle il faut ajouter trois qui est la profondeur maximale des facteurs de ce produit. On voit que chaque solution de $C(\underline{x},\underline{y}) = 1$ s'étend de manière unique à une solution de $T(\underline{x},\underline{y},\underline{z}) = 1$, et que chaque solution de $T(\underline{x},\underline{y},\underline{z}) = 1$ est extension d'une solution de $C(\underline{x},\underline{y}) = 1$.

En particulier, l'équation $C(\underline{x},\underline{y}) = 1$ est solvable si et seulement si l'équation $T(\underline{x},\underline{y},\underline{z}) = 1$ l'est aussi. **Fin**

Le 0 et le 1 entiers, et en fait le 0 et le 1 de n'importe quel anneau, peuvent simuler le 0 et le 1 booléen de la manière suivante. On pose $x \otimes y = x.y$, $x \oplus y = x + y - 2.x.y = x + y + (-1).x.y + (-1).x.y$, qui ont bien les valeurs attendues lorsque x et y ne prennent que les valeurs 0 ou 1. En remplaçant, dans un circuit arithmétique binaire de complexité c, la somme et le produit modulo deux par leurs valeurs ci-dessus, on obtient un circuit de complexité inférieure à $5.c$ qui calcule la même fonction si on restreint ses variables à l'ensemble $\{0,1\}$. En conséquence, toute suite de P se simule ainsi par une suite de $VPdl(-1)$.

Si on fait cette substitution dans un terme booléen, on n'obtient pas directement un terme arithmétique, car dans chaque expression de $x \oplus y$ comme terme l'une au moins des variables apparaît deux fois ; il est cependant exact qu'une suite de Pt se représente par une suite de $VPt(-1)$: c'est une conséquence de la parallélisation des termes, et du fait que la simulation multiplie la profondeur par quatre. C'est même la seule chose qui garantisse une simulation de degré borné, puisque le produit intervient dans l'expression de $x \oplus y$: la simulation ne conserve ni le degré formel, ni la notion de circuit multiplicativement disjoint ! On observe que le terme du lemme précédent est parallélisé d'emblée.

Une question apparemment plus modeste que $P = ? Pt$, c'est de se demander si toute suite de P se simule dans $VPmd(-1)$, ou dans $VPdb(-1)$.

Par exemple, c'est un problème facile à résoudre que de déterminer si une matrice booléenne est une matrice de permutation, si bien qu'il est possible de simuler la la suite de fonctions associée par un élément de $VPdl(-1)$; pour avoir des termes, on doit paralléliser l'algorithme, ce qui revient à trouver un polynôme astucieux comme nous l'avons fait, qui en quelque sorte décrit de manière déclarative la situation (c'est sur ce même principe qu'est fondé la démonstration du lemme ci-dessus).

Cette simulation permet d'établir un lien entre les questions $P = ? NP$ et $VP = ? VSP$:

Lemme 12.2 (IMPLAUSIBILITE DE VP = VSP). *Chacune des égalités* VPdl(-1) = VSPdl(-1) , VPdb(-1) = VSPdb(-1) , VPmd(-1) = VSPmd(-1) , VPt(-1) = VSPt(-1) , VPdl(Z) = VSPdl(Z) , VPmd(Z) = VSPmd(Z) , VPt(Z) = VSPt(Z) *implique* P = P# *et* P = NP *(version non-uniforme)*.

Démonstration. Chacune de ces égalités implique que VSPt(-1) est inclus dans VPdl(Z) , et il suffit de voir que cette condition implique P = P# , qui, comme nous l'avopns remarqué, implique P = NP . Comme NP = NPt , une suite de fonctions $f_n(\underline{x})$ dans NP provient d'une suite de fonctions $g_n(\underline{x},\underline{y})$ dans Pt ; cette dernière a une simulation $h_n(\underline{x},\underline{y})$ dans VPt(-1) , dont la somme de Valiant $k_n(\underline{x}) = \Sigma_{\underline{y}\ \text{booléen}}\ h_n(\underline{x},\underline{y})$ est dans VSPt(-1) , donc dans VPdl(Z) par hypothèse. Lorsqu'on donne à $\underline{x}$ des valeurs booléennes, $k_n(\underline{x})$ est égal au nombre de solutions $\underline{y}$ de l'équation $g_n(\underline{x},\underline{y}) = 1$. Un calcul numérique dans VPdl(-1) ne peut se suivre en chiffres, car il peut donner des nombres demandant un nombre exponentiel de chiffres, et c'est bien pire dans VPdl(Z) , où l'on se permet d'utiliser comme constantes des entiers de taille imprévisible. Mais dans le cas d'espèce nous calculons un nombre compris entre 0 et 2^m , où m est la longueur de $\underline{y}$, et on peut se contenter de faire le calcul modulo 2^{m+1} , chaque nombre intermédiaire se représentant par m+1 chiffres ; vérifier que $g_n(\underline{x},\underline{y}) = 1$ a des solutions revient à vérifier que $k_n(\underline{x})$ n'est pas nul, c'est-à-dire a au moins un chiffre égal à 1 : tout cela représente clairement un calcul booléen de complexité polynomiale, se traduisant par une suite de circuits arithmétiques modulo deux de taille polynomiale. **Fin**

Nos hypothèses VP = VSP impliquent quelque chose d'encore plus invraisemblable que P = NP , puisqu'elles nous permettent d'avantage que de seulement déterminer si l'équation $g_n(\underline{x},\underline{y}) = 1$ a des solutions ou pas : elles rendent possible le calcul du nombre exact de ses solutions. Cette dernière hypothèse est connue sous le nom de P = #P .

13. La fonction-coefficient.

Nous considérons ici des objets plus généraux que des polynômes, que nous appelons fonctions-polynômes, qui à un uple $\underline{y}$ de $\{0, 1\}^m$ associe un polynôme de $K[\underline{x}]$. Nous les notons $f(\underline{x};\underline{y})$, étant entendu que les variables y_i ne prennent que les valeurs 0 ou 1 (nous dirons que ce sont des variables booléennes). Il est possible que $\underline{x}$ soit de longueur nulle, et qu'on ait alors une fonction à valeurs dans K .

Nous observons tout d'abord qu'à toute fonction-polynôme $f(\underline{x};\underline{y})$ est associé un polynôme $F(\underline{x};\underline{y})$, utilisant, en plus de -1 , les mêmes constantes que f , tel que pour tout uple $\underline{e}$ de $\{0,1\}^m$ le polynôme $f(\underline{x};\underline{e})$ soit le polynôme $F(\underline{x};\underline{e})$. Ce

polynôme F n'est bien sûr pas unique, puisque seules comptent les valeurs qu'il prend lorsque les y_i valent 0 ou 1 .

L'existence de F , évidente lorsque m = 0 , se montre par induction sur m . Vérifions le passage de m à m+1 : par hypothèse d'induction f(x;y,0) se représente par un polynôme $F_0(x;y)$, et f(x;y,1) par un polynôme $F_1(x;y)$; et alors $f(x;y,z) = z.F_1(x;y) + (1-z).F_0(x;y)$.

Nous appellerons complexité de la fonction-polynôme f(x;y) la complexité minimale d'un polynôme F(x;y) qui ainsi la représente. Nous pouvons définir des classes VP , VSP , etc., pour les suites de fonctions-polynômes comme nous l'avons fait pour les suites de polynômes.

La principale fonction-polynôme que nous allons considérer, c'est la fonction-coefficient d'un polynôme. Nous l'avons déja fait pour le permanent, dont nous avons vu que la fonction-coefficient est particulièrement simple (il en est de même de celle du déterminant !).

Si f(x) est un polynôme en n variables de degré au plus d , pour représenter les monômes on écrira en base deux le degré d_i du monôme par rapport à chaque inconnue x_i , ce qui demande un uple y_i de longueur au plus log(d) + 1 . Autrement dit la fonction-coefficient g(y) de f(x) dépend d'au plus n.(log(d) + 1) variables booléennes $y = (y_1, \dots y_n)$. Sa fonction-coefficient est la fonction Cf(y) qui à y booléen associe le coefficient du mônôme de f de multidegré y . On peut également considérer la fonction monôme Mf(x,y) , qui au uple booléen y associe le monôme de f de multidegré y .

La fonction-coefficient définie ci-dessus est la fonction-coefficient totale du polynôme f(x) . On peut définir des fonctions-coefficients partielles en séparant le uple de variables en deux blocs x' et x" , et en développant le polynôme seulement par rapport au premier : les coefficients sont alors des polynômes en x" . On peut aussi définir les fonctions-coefficients, partielles ou totales des fonctions-polynômes.

Dès qu'on dispose de la constante -1 , les complexités de la fonction-coefficient et de la fonction-monôme sont polynomialement liées. D'une part Cf(x,y) = Mf(1,y) . D'autre part Mf(x,y) = Cf(y).(x^y) , où x^y est le produit des x_i^y_i qui sont ainsi définis : si $y = (y_1, \dots y_{1+\log(d)})$, x^y est le produit des polynômes $y_i(x^{2^i} - 1) + 1$, valant 1 si $y_i = 0$ et x^{2^i} si $y_i = 1$. Ce polynôme x^y est calculé par un circuit de complexité 5n.(1+log(d)) .

La complexité de x^y comme terme est bornée par n(d+5) , si bien que, si le degré de f est polynomialement borné, ce sont même les complexités de ces deux fonctions en tant que termes qui sont polynomialement liées.

Le lemme suivant a pour but de décrire la fonction-coefficient du polynôme calculé par un circuit multiplicativement disjoint. Rappelons qu'un monôme unitaire est un monôme de coefficient 1 , et qu'un polynôme calculé par un circuit arithmétique sans paramètres est somme de monômes unitaires, ses monômes étant obtenus en regroupant les monômes unitaires de même multidegré. Ce lemme suit la trace des monômes unitaires dans le circuit.

Lemme 13.1 (LOCALISATION DES MONOMES). *Soit* $C(\underline{x})$ *un circuit multiplicativement disjoint, sans constantes, de complexité* c *, de degré (formel complet)* d *, en* n *variables* $\underline{x}$ *. Il existe un circuit* $M(\underline{x},\underline{z})$ *, utilisant seulement la constante* -1 *, de complexité inférieure à* $c(c+4)$ *, de degré formel complet inférieur à* $d + 2c$ *, dont les variables* $\underline{z}$ *sont indexées par les portes d'addition de* C *, et tel que si* $\underline{z}$ *est un uple booléen,* $M(\underline{x},\underline{z})$ *calcule le polynôme nul, ou bien un monôme unitaire en* $\underline{x}$ *, chaque monôme unitaire de* C *étant représenté une fois et une seule de cette manière. Par ailleurs, le multidegré de ces monômes est calculé par un multicircuit* $D(\underline{z})$ *, utilisant* -1 *comme seule constante, de complexité inférieure à* $n(4.(c+1)^2 + c^A)$ *, et de degré formel complet inférieur à* $2c.c^B$ *, où* A *et* B *sont des constantes facilement calculables.*

Démonstration. Lorsqu'on arrive à une porte d'addition p , recevant ses flèches des portes p' et p'' , éventuellement confondues, les monômes unitaires qui arrivent en p proviennent soit de p' , soit de p'' ; tandis que s'il s'agit d'une porte de multiplication, les monômes unitaires de p sont produits d'un monôme unitaire de p' et d'un monôme unitaire de p'' .

Pour construire le circuit M , nous introduisons des portes d'entrée M pour les variables z , indexées par les portes d'addition de C , et pour chaque porte p de C une porte $\varphi(p)$ qui, lorsque $\underline{z}$ est booléen, calculera le monôme arrivant en p qui est associé à la restriction de $\underline{z}$ au variables correspondant à des portes d'addition du sous-circuit défini par p , plus quelques autres portes pour les calculs intermédiaires.

Si p est une entrée associée à la variable x , $\varphi(p)$ également.

Si p est une porte de multiplication des portes p' et p'' , comme C est multiplicativement disjoint, le uple de variable codant les monômes unitaires de p est la concaténation de ceux associés à p' et à p'' ; $\varphi(p)$ multipliera donc $\varphi(p')$ et $\varphi(p'')$.

Si p est une porte d'addition des portes p' et p'' , elle bénéficie d'une nouvelle variable z , que nous utilisons pour choisir entre p' et p'' . Le polynôme $z.u + (1-z).v$ vaut u si $z = 1$, et v si $z = 0$. Mais il ne faut pas sélectionner directement entre $\varphi(p')$ et $\varphi(p'')$, car s'il y a des variables dans $\underline{z}$ associées à p'' mais pas à p' , cela ferait apparaître plusieurs fois les monômes de $\varphi(p')$. Pour corriger cela, il faut forcer toutes ces variables à valoir 1 pour que le monôme apparaisse ; si donc Z' représente le produit des variables correspondant à une porte d'addition utilsée par p' et pas par p'' , et Z'' le produit des variables correspondant à une porte d'addition utilisée par p'' et pas par p , on pose $\varphi(p)$

= z.Z'.φ(p') + (1-z).Z".φ(p") . Remarquons en passant que, si p' = p" , φ(p) = φ(p') , mais que chaque monôme est bien compté deux fois car il y a deux valeurs possibles pour z .

Le nombre de variables figurant dans z , Z et Z' est inférieur à c , si bien que le calcul associé à une porte d'addition demande moins de c + 4 portes, d'où la borne sur la complexité de M .

Pour le degré formel, nous montrons par induction sur la profondeur que celui de φ(p) est borné par d + 2a , où d est le degré formel de p et a le nombre de portes d'addition du sous-circuit qui lui est associé. Ca va si p est une entrée. Si p est une multiplication, le degré formel de φ(p) est majoré par d' + 2a' + d" + 2a" , qui vaut d + 2a puisque le circuit est multiplicativement disjoint. Si p est une addition, le degré formel de z.Z'.φ(p') est borné par d' + a' + a < d' + 2a , et celui de (-1).(z).Z".φ(p") par d" + a + a" + 1 ≤ d" + 2 a , si bien que le degré de φ(p) est bien majoré par d + 2a .

Passons maintenant à la construction d'un multicircuit D(z) qui, lorsqu'on lui donne en entrée un uple booléen z qui représente un monôme unitaire de C , donne en sortie le multidegré de ce monôme ; ce qu'il fait pour les autres valeurs booléennes de z , telles que M(x,z) = 0 , n'a pas d'importance. On calcule le degré par rapport à une variable x , et ensuite on reproduit n fois le multicircuit obtenu ; pour exprimer ce degré, il faut faire une sélection quand on franchit une porte d'addition, et simuler une addition binaire quand on franchit une porte de multiplication. Les sélections n'ont pas d'influence catastrophique sur le degré formel, mais une accumulation d'additions de nombres écrits en base deux, même si elles sont parallélisées, le fera exploser. Pour éviter cela, on représente le degré "en bâtons", c'est-à-dire par le nombre de 1 d'un uple booléen ; l'addition est alors une simple juxtaposition, qui ne demande aucun calcul. C'est seulement à la fin qu'on collecte les bâtons pour exprimer leur nombre en base deux.

On procède ainsi. Pour chaque porte p de C , on introduit un uple ψ(p) de c(p)+1 portes , où c(p) est la complexité du sous-circuit de C défini par p . Si p est une entrée, la porte de ψ(p) calcule 1 ou 0 suivant que la variable associée à p est x ou non ; comme ces calculs se font à partir de -1 , les entrées demandent au plus 2n opérations, et un degré formel 2 . Si p est une porte de multiplication, ψ(p) est la réunion de ψ(p') et de ψ(p") , ce qui fait bien c(p') +1 + c(p") + 1 = c(p) + 1 portes, puisque le circuit est multiplicativement disjoint : on n'a même pas besoin d'ajouter de nouvelles portes ! Si p est une porte d'addition, on sélectionne grâce à la variable z qui lui est associée : ψ(p) = z. ψ(p') + (1-z).ψ(p") , les uples ψ(p') et ψ(p") ayant été complétés par des 0 pour atteindre la longueur de ψ(p) . La porte d'addition p augmente le degré formel de deux, et demande la création de moins de 3.c(p) + 2 nouvelles portes. Comme, dans un sleuppe, la i-ème porte ne suit pas plus de i opérations, le nombre de portes nécessaires jusqu'à présent est majoré par $2.n + \Sigma_1^c (3.i + 2) = 2.n + 2.c + (3/2).c.(c+1) < (c+1).((3/2).c + 4) \leq 4.(c+1)^2$; quand au degré formel, il est majoré par 2c .

Quand on a fini, il faut compter tout les 1 qu'on trouve dans le ψ de la sortie, et exprimer leur nombre en base deux ; pour préserver le degré formel, il faut le faire en parallèle, ce qui est le petit cochon algorithmique (voir les additions d'Ofman et l'algorithme de Jules et Jim dans les Petits Cailloux). Leur simulation par des polynômes demandera c^A opérations et se fera en profondeur $B.\log(c)$, ce qui multipliera le degré formel par c^B. **Fin**

Lemme 13.2 (STABILITE DE VSPmd PAR PRISE DE COEFFICIENTS). *Si* A *contient* -1 *(i.e. le* -1 *de son anneau), une suite de polynômes est dans* VSPmd(A) *si et seulement si sa fonction-coefficient totale y est, si et seulement si chacune de ses fonctions-coefficients partielles y est, si et seulement si l'une de ses fonctions-coefficients partielles y est.*

Démonstration. Si f_n est une suite de VPmd(∅) dont les monômes unitaires sont décrits par la suite de circuits $M_n(\underline{x},\underline{z})$, pour obtenir le circuit $M_n(\underline{x},\underline{y},\underline{z})$ qui ne conserve que les monômes de multidegré $\underline{y}$, il suffit de le multiplier par le produit des $1 - u_i - y_i + 2.u._i y_i$ où $\underline{u}$ est le uple calculé par le multicircuit $D_n(\underline{z})$. D'après le lemme précédent, la suite de polynôme définie par $M_n(\underline{x},\underline{y},\underline{z})$ est dans VPmd(-1), et comme la fonction coefficient de f_n est donnée par $\Sigma_{\underline{z}\ \text{booléen}}\ M_n(\underline{1},\underline{y},\underline{z})$, elle est dans VSPmd(-1). Cela vaut aussi bien pour les fonctions-coefficients partielles, puis pour les suites f_n de VSPmd(∅) par associativité des sommes de Valiant, et aussi de VSPmd(A) par substitution de constantes.

Réciproquement, si la fonction coefficient $c_n(\underline{y})$, partielle ou totale, de $f_n(\underline{x})$ est dans VSPmd(A), la formule $f_n(\underline{x}) = \Sigma_{\underline{y}\ \text{booléen}}\ c_n(\underline{y}).(\underline{x}\hat{}\underline{y})$ montre que f_n est dans VSPmd(A). **Fin**

En augmentant le nombre de variables dans la sommation, et en procédant de façon plus déclarative et moins calculatoire, c'est-à-dire de façon globale et non pas pas-à-pas, on peut remplacer les circuits par des termes dans le lemme précédent, ce qui donne le résultat suivant, analogue de NP = NPt, qui est essentiel dans les travaux de Valiant. La démonstration donnée ci-dessous est de Malod : elle est considérablement plus claire - et plus sûre - que celle donnée primitivement par Valiant.

Lemme 13.3 (EQUIVALENCE DES TERMES ET DES CIRCUITS SOUS UNE SOMME DE VALIANT). *Si* A *contient* -1 *,* VSPmd(A) = VSPt(A).

Démonstration. Considérant un circuit $C(\underline{x})$ multiplicativement disjoint sans constantes, nous observons que ses monômes unitaires correspondent à ses sous-graphes M satisfaisant aux conditions suivantes : (i) la sortie est dans M (ii) si une flêche est dans M, il en est de même des deux portes qu'elle joint (iii) toute

porte de M , à l'exception de la sortie, émet une flêche dans M (iv) si une porte de multiplication est dans M , il en est de même des deux flêches qu'elle reçoit (v) si une porte d'addition est dans M , elle reçoit exactement une flêche dans M . Comme le graphe est multiplicativement disjoint, ce sous-graphe a une structure arborescente, et le monôme qu'il calcule est le produit des inconnues figurant à ses entrées.

Pour représenter un tel sous-graphe par un uple booléen, on introduit une variable z_p par porte de C et une variable z_f par flêche de C . A chacune des conditions ci-dessus on associe des polynômes qui valent 1 pour les uples booléens qui les satisfont, et 0 pour ceux qui ne les satisfont pas. Ce sont : (i) z_s , où s est la porte de sortie ; (ii) $1 - z_f + z_f.z_p.z_{p'}$ où la flêche f part de p et arrive en p' ; (iii) $1 - z_p + z_p.P_k(z_1, \dots z_k)$, où $z_1 \dots z_k$ représentent les flêches qui partent de la porte p , et où le polynôme P est défini par la récurrence suivante : $P_1(z_1) = z_1$, $P_k(z_1, \dots z_k) = z_k + (1 - z_k).P_k(z_1, \dots z_{k-1})$; (iv) $1 - z_p + z_p.z_f.z_{f'}$, où la porte de multiplication p reçoit les deux flêches f et f' ; (v) $1 - z_p + z_p.(z_f + z_{f'} - 2.z_f.z_{f'})$, où la porte d'addition p reçoit les deux flêches f et f' . A chaque porte d'entrée p , on associe le polynôme $1 - z_p + z_p.x$, où x est la variable qui étiquette p .

Le produit de tous ces polynômes est un terme $T(\underline{x},\underline{z})$ de taille inférieure à $1 + 6.2.c + 3.(c+1)^2 + 8.c + 5.(c+1)$. Pour $\underline{z}$ booléen, $T(\underline{x},\underline{z})$ calcule le polynôme nul, ou bien un monôme unitaire de C , dont tous les monômes unitaires sont ainsi bijectivement représentés ; $C(\underline{x})$ s'obtient donc par somme de Valiant devant $T(\underline{x},\underline{z})$. Cela montre que $VPmd(\varnothing)$ est inclus dans $VSPt(-1)$, puis le lemme par substitution de constantes et associativité des sommations.

En ajoutant encore des variables, on peut obtenir simplement la fonction-coefficient de $C(\underline{x})$ comme somme de Valiant devant un terme de taille polynomiale, ce qui donne une nouvelle démonstration du lemme précédent. Pour cela, il faut introduire de nouvelles variables $\underline{v}(p)$ représentant les multidegrés du monôme lorsqu'il passe par la porte p . Le uple de variable $\underline{y}$ est associé à la sortie.

On construit un terme $D(\underline{x},\underline{y},\underline{z},\underline{v})$ en prenant le produit des polynômes suivants : si p est une entrée, étiquetée par la variable x , et v est la variable associée à son degré en x , on met dans le produit le facteur $1 - z_p + z_p.v$; si au contraire w est la variable associée au degré de p par rapport à une variable autre que x , on met $1 - w$; si p est une porte d'addition, recevant les flêches f' et f'' des portes p' et p'' , on met le produit des $v_i - z_{f'}.v_i' - z_{f''}.v_i''$, où v_i , v_i' et v_i'' parcourent les uples de variables représentant les multidegrés respectifs de p , p' et p'' ; si p est une porte de multiplication, il faut mettre le produit des $v_i - s_i(\underline{v}',\underline{v}'')$, où $s(\underline{v}',\underline{v}'')$ représente une simulation de l'addition en base deux de $\underline{v}'$

et de $\underline{v}$" ; comme les longueurs de ces uples sont de l'ordre de $\log(c)$, il n'est même pas nécessaire de paralléliser cette addition. Pour éviter que le même monôme soit compté plusieurs fois, il faut mettre aussi le produit des $z(p) + (1 - z(p)).v$ pour chaque variable v associée à la porte p .

Lorsque $\underline{y}$, $\underline{z}$ et $\underline{v}$ sont booléens, et que $D(\underline{x},\underline{y},\underline{z},\underline{v})$ ne calcule pas le polynôme nul, c'est que $\underline{z}$ représente un monôme unitaire de $C(\underline{x})$, dont $\underline{y}$ est le multidegré, les autres valeurs de $\underline{v}$ représentant le multidegré du monôme en formation à la porte p si elle a été utilisée, étant égales à 1 sinon. La fonction-monôme de $C(\underline{x})$ s'exprime comme la somme de Valiant $\Sigma_{\underline{z},\ \underline{v}\ \text{booléens}}\ T(\underline{x},\underline{z}).D(\underline{x},\underline{y},\underline{z},\underline{v})$. **Fin**

14. Polynômes VSP-complets.

On dit que le polynôme $f(\underline{x})$ est projection du polynôme $g(\underline{y})$ si le premier s'obtient à partir du second en remplaçant les variables de ce dernier par des variables ou des constantes. Un célèbre théorème de Valiant affirme qu'il existe un polynôme $p(c,d)$, tel que tout polynôme de complexité c et de degré d soit une projection de $\mathrm{Per}_{p(c,d)}$. Une étape essentielle de la démonstration, c'est de montrer, comme nous l'avons fait, que $\mathrm{VSPmd}(-1) = \mathrm{VSPt}(-1)$; ensuite on fait des calculs dans des graphes ; les constantes qui interviennent dans les substitutions sont 1 , -1 et $1/2$.

Autrement dit, le Permanent est universel (on dit aussi "complet") pour la classe $\mathrm{VSPmd}(-1/2) = \mathrm{VSPt}(-1/2)$: pour toute suite de polynôme $f_n(\underline{x})$ qui est dans $\mathrm{VSPmd}(-1/2)$, il existe un polynôme $p(n)$ tel que, pour chaque n , $f_n(\underline{x})$ s'obtienne à partir de $\mathrm{Per}_{p(n)}$ en y remplaçant des variables par des variables ou l'une des constantes $-1 = (-1/2) + (-1/2)$, $1 = (-1).(-1)$ ou $-1/2$. Comme par ailleurs le Permanent est dans $\mathrm{VSPt}(-1/2)$, il est clair que l'hypothèse $\mathrm{VSPmd}(-1/2) = \mathrm{VPmd}(-1/2)$ équivaut à l'appartenance du Permanent à la classe $\mathrm{VPmd}(-1/2)$: le Permanent est en quelque sorte l'élément typique de la classe $\mathrm{VSPmd}(-1/2)$.

Pour une formulation plus valiantesque, on affaiblit ce résultat en disant que le Permanent est complet pour la classe $\mathrm{VSPdb}(Q) = \mathrm{VSPmd}(Q) = \mathrm{VSPt}(Q)$, que $\mathrm{VSPdb}(Q) = \mathrm{VPdb}(Q)$ si et seulement si le permanent est dans $\mathrm{VPdb}(Q)$.

En caractéristique $p \neq 2$, la constante $-1/2$ s'inverse modulo p , si bien que le Permanent est complet pour la classe $\mathrm{VSPdb}(\mathrm{mod}\ p)$, qui est aussi la classe $\mathrm{VSPmd}(\mathrm{mod}\ p)$. Il est par contre très peu probable que le permanent soit complet pour la classe $\mathrm{VSPmd}(\mathrm{mod}\ 2)$, car il est facilement calculable modulo 2 , et le Théorème de Valiant n'affirme nullement cette complétude.

Pour avoir un polynôme universel quelle que soit la caractéristique, Malod a introduit le Hamiltonien Ham_n , défini de manière analogue au permanent, mais

avec une somme restreinte au (n-1)! permutations circulaires des colonnes. Il montre que le Hamiltonien est dans VSPmd(-1) , et est universel pour cette classe ; les seules constantes à substituer sont 1 et -1 , si bien que le Hamiltonien reste universel pour la classe VSPmd(mod p) même si p = 2 .

Enfin, Malod a introduit un polynôme (ou plutôt une suite de polynômes !) Mal_n qui est universel pour la classe VSPdl(-1) ; il est assez gigantesque, mais sa fonction-coefficient n'est pas trop compliquée.

15. Des résultats de nature hypothétique, en degré borné

La philosophie qu'on peut tirer de la section 12, qui est celle que j'ai donnée à Malod comme sujet à étudier pour sa thèse, c'est qu'une hypothèse VPdb = VSPdb revient à dire qu'on passe d'un polynôme à sa fonction-coefficient, et réciproquement, sans explosion de complexité. Nous en explicitons deux interprétations, la deuxième étant un relâchement de la première.

THEOREME 15.1. *Les improbables hypothèses suivantes sont équivalentes :*
(i) VSPmd(-1) = VPmd(-1) *;*
(ii) *le Hamiltonien est dans* VPmd(-1) *;*
(iii) *si* f_n *est dans* VPmd(-1) *, toutes ses fonctions-coefficient partielles aussi ;*
(iv) *si la fonction-coefficient de* f_n *est dans* VPmd(-1) , f_n *aussi ;*
(v) *si on dérive un élément de* VPmd(-1) *par rapport à plusieurs de ses variables, on obtient un élément de* VPmd(-1) .

Démonstration. Pour (ii), c'est parce que le Hamiltonien est universel. Pour (iii) , c'est parce que VSPmd(-1) est stable sous la prise de coefficients ; pour la réciproque, nous avons vu qu'un polynôme facile à calculer a un coefficient égal au permanent, et on peut faire la même chose avec le Hamiltonien. Pour (iv), c'est parce qu'un polynôme s'obtient par somme de Valiant devant sa fonction-monôme. Pour (v), on différencie un polynôme en calculant les dérivées de chacun des monômes, puis on somme : les deux passages se font sans explosion de complexité ; pour la réciproque, nous avons vu un polynôme facile à calculer a pour dérivée itérée le permanent, et il est possible de donner un exemple de même nature avec le Hamiltonien. **Fin**

Le même théorème vaut naturellement pour VPdb(mod p) = VSPdb(mod p) .

THEOREME 15.2. *Les improbables hypothèses suivantes sont équivalentes :*
(i) VSPdb(Q) = VPdb(Q) *;*
(ii) *le Hamiltonien (ou le Permanent) est dans* VPdb(Q) *;*
(iii) *si* f_n *est dans* VPdb(Q) *, toutes ses fonctions-coefficient aussi ;*
(iv) *si la fonction-coefficient de* f_n *est dans* VPdb(Q) , f_n *aussi ;*

(v) *si on dérive un élément de* VPdb(Q) *par rapport à plusieurs de ses variables, on obtient un élément de* VPdb(Q) ;
(vi) *si on intègre un élément de* VPdb(Q) *par rapport à plusieurs de ses variables, on obtient un élément de* VPdb(Q) .

Démonstration. Tout se démontre de la même façon ; pour (iv) , il y a des coefficients rationnels qui interviennent dans l'intégration des monômes, ce qui nous avait empêché de parler d'intégration en 14.1. **Fin**

16. Des résultats, de nature hypothétique, en degré libre

En degré libre, un problème ouvert majeur est de savoir si la classe VSPdl(-1) est stable pour la prise de coefficients. Dans ce cas, il y a un nombre doublement exponentiel de monômes unitaires, et il n'est plus posible de les localiser individuellement ; c'est parce qu'il n'y a qu'un nombre simplement exponentiel de multidegrés possibles qu'ils se regroupent en paquets doublement exponentiels. Malod a montré un lemme combinatoire, permettant de calculer les coefficients des paquets à partir du calcul de coefficients binomiaux $C_n^m = (n^m)$, où les nombres n et m sont simplement exponentiels. Il s'agit bien d'un calcul, et non pas du don gratuit de quelques paramètres entiers, car les nombres n et m dépendent de l'architecture du circuit considéré.

Il est peu probable que ce calcul puisse se faire dans VPdl(-1) , car cela impliquerait qu'on puisse calculer n! en un nombre polynomial en log(n) de sommes et de produits, ce qui donnerait d'une part un test simpliste de primalité, et d'autre part un algorithme de factorisation, aux conséquences horribles pour la vie de l'homme moderne (et pas seulement pour sa carte bleue ; si vous savez factoriser, vous pouvez téléphoner à Poutine : "Allo, Vlady, c'est assommant, mais on s'est trompé de bouton ; les bombes arrivent sur Moscou dans vingt minutes et on sait pas quoi faire pour les arrêter !" avec signature George W. Bush authentifiée). Mais il pourrait se faire dans VSPdl(-1) .

Faute de ce résultat, on ne peut énoncer d'analogues aux théorèmes de la section 14 correspondant à l'hypothèse VPdl(-1) = VSPdl(-1) . Mais, comme l'a observé Malod - toujours lui - un vieux (XIX° siècle) théorème de Lucas permet le calcul rapide des coefficients binomiaux modulo p , pour chaque nombre premier p fixé. Cela nous permet de conclure par un dernier théorème, dont la dernière clause, due bien sûr à Malod, est assez sensationnelle :

THEOREME 16.1. *Les improbables hypothèses suivantes sont équivalentes :*
(i) VSPdl(mod p) = VPdl(mod p) *;*
(ii) *si* f_n *est dans* VPmd(mod p) *, toutes ses fonctions-coefficient aussi ;*
(iii) *si la fonction-coefficient de* f_n *est dans* VPmd(mod p) *,* f_n *aussi ;*
(iv) *si on dérive un élément de* VPmd(mod p) *par rapport à plusieurs de ses variables, on obtient un élément de* VPmd(mod p) *;*
(v) VSPdb(mod p) = VPdb(mod p) .

Démonstration. Tout est clair, sauf l'équivalence de (i) et de (v) ; on passe de (i) à (v) par troncature du degré ; pour l'autre sens, en examinant la fonction-coefficient du polynôme universel Mal_n on voit qu'elle est dans VSPdb(mod p) , si bien qu'un polynôme de VSPdl(mod p) s'obtient en remplaçant dans un polynôme de VSPdb(mod p) certaines inconnues par des monômes de haut degré. **Fin**

En caractéristique 0 , la troncature montre que VSPdl(Z) = VPdl(Z) implique VSPdb(Z) = VPdb(Z) , mais quid de la réciproque ? Encore plus mystérieux sont les rapports entre VSPmd(-1) = VPmd(-1) et VSPdl(-1) = VPdl(-1) .

SUMMARY OF THE SEVEN LAST SECTIONS

Impossible problems

1. Show that $P \neq Pt$, $P \neq NP$, $P \neq P\#$.
2. Show that the determinant is in VPt(-1) ; show that the permanent is not in VPdb(-1) , an even not in VPdb(C) .
3. Call $pascaline_n$ the function from $\{0,1\}^{2n}$ to N which associates to $(\underline{x},\underline{y})$ the binomial coefficient $\binom{a}{b}$, where a and b are the integers whose binary developments are $\underline{x}$ and $\underline{y}$; show that the pascaline is not in VPdl(-1) .
4. Call $factorial_n$ the function from $\{0,1\}^n$ to N which associates to $(\underline{x})$ the number a! , where a is the number with binary development $\underline{x}$; show that the factorial is not in VPdl(-1) .

Difficult problems

5. Show that $P \neq Pmd$, $P \neq Pc$.
6. Can any computation in P be simulated in VPmd(-1) ?
7. Show that VPt(-1) ≠ VPmd(-1) , VSPt(-1) ≠ VPmd(-1) , VPdl(-1) ≠ VSPdl(-1) .

Problems

8. Is the class VPdl(-1) closed under taking coefficient-functions ?
9. Does the pascaline belongs to VSPdl(-1) , or VSPt(-1) ?

10. Does the factorial belongs to VSPdl(-1) , or VSPt(-1) ?
11. Can you show that the two hypothesises VPdl(-1) ≠ VSPdl(-1) and VPmd(-1) ≠ VSPt(-1) are equivalent ?

Easy problem

12. Learn French.